REPORT DOCUMENTATION PAGE

AFRL-SR-AR-TR-05-0246

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
08-06-2005	Technical Report - Final Report	01-09-2004 to 31-05-2005

4. TITLE AND SUBTITLE
StegKit: Automated Steganalysis Tool

Final Report

5a. CONTRACT NUMBER
FA9550-04-C-0109

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Robert J. Bechtel

Mike Rowe

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Eclectic Computing Concepts
1650 W. Virginia Street
Suite 200
McKinney, TX 75069-7703

University of Wisconsin - Platteville
1 University Plaza
Platteville, WI 53818-3099

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
4015 Wilson Blvd, Room 713
Arlington, VA 22203-1954

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Report developed under STTR contract for topic AF04-T008. There are many different techniques for hiding steganographic content and in theory, any file type can serve as a carrier. We developed a general-purpose steganalysis system that would handle many file types and many steganographic methods, while being easy to configure and use. We created a flexible, efficient framework, called *StegKit*, into which steganalysis components could be placed or "plugged in" to carry out automated and manual analysis within common operating environments. The system was modeled on modern antivirus software, capable of background monitoring of files, email, and web pages visited in a browser. Our effort demonstrated the feasibility of a technical architecture featuring a centralized controller with plug-in components for file type detection and steganalysis.

15. SUBJECT TERMS

STTR Report, steganalysis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (include area code)
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	141	

Standard Form 298 (Rev. 8-98)

StegKit Phase I STTR (FA9550-04-C-0109)
Final Report

Eclectic Computing Concepts and University of Wisconsin - Platteville

Table of Contents

Project Objectives 1
Work Carried Out 3
Results Obtained 11
Estimate of Technical Feasibility 14
Cumulative List of Persons Involved 15
Publications 15
Appendix A - StegKit Final Phase I Release Notes 16
Appendix B - Source Code Listings 33

Project Objectives

Surreptitious communication through steganography has application in many settings. There are many different techniques for hiding steganographic content and in theory, any file type can serve as a carrier. Most studies in steganalysis have focused on single file types and a few steganographic methods on those file types, representing a limited sample of the population of transmitted data. In our Phase I effort, we proposed to develop a general-purpose steganalysis system that would handle many file types and many steganographic methods, while being easy to configure and use. Such a system could be deployed widely enough to adequately sample a statistically significant portion of transmitted data. A distinguishing feature of our proposed approach was the use of machine learning techniques to create a file type classifier that can be used to select among available steganalysis modules. The resulting system was to be modeled on modern antivirus software, capable of background monitoring of files, email, and web pages visited in a browser.

Our plan was to create a flexible, efficient framework, called *StegKit*, into which steganalysis components could be placed or "plugged in" to carry out automated and manual analysis within common operating environments. Specific steganalysis algorithms and techniques, while important for demonstrating feasibility and the concept of operations, were a lesser concern in Phase I, but were anticipated to be a the primary focus of Phase II. The framework to be created in Phase I was to provide a stable and well-defined platform for rapid testing, evaluation, and evolution of steganalysis components in Phase II. In Phase I, we proposed to develop for deployment under the Microsoft Windows operating system (either Windows 2000 or Windows XP).

Existing steganalysis tools, such as *stegdetect* [24], tend to be structured as utility programs, invoked from the command line and applied to suspect files one at a time.

While effective for checks of a small number of suspect files, they are not packaged in a way that supplies their services in a continuous background fashion covering, for example, all material accessed through a web browser. The existing tools also tend to focus on a single cover file type (e.g., image files in JPEG encoding), thus requiring the user to select the appropriate utility for maximum efficacy.

Some work reported in the literature [22] has adapted command line utilities to work in a "batch" mode against suspect files collected by a separate web spider process. This technique has been used to examine large numbers of image files from EBay and Usenet newsgroups as a means of assessing the prevalence of steganographically concealed material on the Internet. The batch approach, while effective for the purpose of the research effort, is not well suited to the dynamic environment of everyday PC use. Furthermore, it also restricted the analysis to a single file type, also inappropriate for use in a dynamic setting.

As noted in the original STTR topic description, a more appropriate packaging model would be computer virus checking. Virus checkers typically can be used under manual control, scanning anything from the current contents of memory, through a single file, all the way to all drives on a computer. They can also usually be configured to run automatically, unobtrusively examining every file as it is created, or every new email as it is received from the mail server. (Some checking software runs on the mail server itself, checking before messages are passed to the mail client.) The specific file type is irrelevant to the virus checking process, though some file types known to compress or otherwise package content (such as zip files or Microsoft cabinet files) may be expanded before being checked. Users frequently have some control over the action taken when a virus is detected, ranging from simple reporting through "quarantining" where the infected file is isolated in a protected area of the directory structure, through to deletion. In some cases, the virus analysis may produce sufficient information to permit disinfection, or removal of the viral component and salvaging of the original, uninfected file.

With this in mind, our objective in Phase I was to demonstrate the feasibility of creating a steganalysis tool that packages steganography detection in a manner similar to that of a virus checker. The functionality of this tool was to be partitioned between a framework and plug-ins, with the plug-ins providing specific steganalytic techniques and the framework providing necessary support services including file handling and user interface. The complete tool, consisting of the framework plus a plug-in based on existing steganalysis components, was to:

- automatically identify all newly created files on the subject computer, including file type,
- identify all newly received email messages on the subject computer, including file type for any attachments,
- perform appropriate steganalysis on the new material, based on the file type,
- report, in a form selected by the user, whenever steganographic content is detected in analyzed material,

- offer a means for the user to manually invoke steganalysis on one or more selected files, and
- offer a user interface for configuration of the analysis process (e.g., selecting target directories, etc.).

In addition to the proof-of-concept demonstration tool, important proposed objectives of Phase I included:

- identifying, documenting, and implementing methods for rapid file type determination,
- estimating acceptable performance bounds for background steganalysis (time and space),
- designing and documenting an application programming interface (API) for plug-in steganalysis modules, including a description of framework services and data structures, and
- establishing a testing environment and executing tests of the concept demonstrator.

Most of these objectives were retained (and achieved) during our Phase I effort. Interaction with the technical advisors at AFRL Rome caused us to de-emphasize file type detection, and as a consequence of this decision, we abandoned our plan to use machine learning techniques to create a reliable, accurate, and high-performing file type detector. We also focused exclusively on file creation and modification, leaving email for later consideration.

Work Carried Out

September 2004

The STTR contract to ECC and subcontract from ECC to UW-P were put in place and effort started at ECC and UW-P. It took a couple of days after the contract was in place to get students hired at UW-P, but this had no negative impact on progress.

We established a project wiki to support inter-site communication, making project documents readily available to all participants. The wiki is password protected to control access to project personnel.

As described in our proposal, the preliminary architecture for StegKit features a controller with plug-ins implemented as Windows DLLs, and a simple API consisting of two functions: init() and detect(). After working with this model, we provisionally added a third function - typeof(). We created some small programs to confirm that basic assumptions underlying the planned architecture, such as the ability to monitor file operations, true dynamic library loading and linking, were valid in the Windows environment. Analysis of the architecture identified some security and performance issues.

We collected information on file type detection and identified three possible approaches: (a) the basic Windows approach, mapping file extensions to application types, (b) the basic Unix approach of magic numbers/bytes, and (c) machine learning from examples.

Finally, we prepared briefing materials for the kickoff meeting scheduled for 07 October at Rome, NY.

October 2004

The kickoff meeting was held at Rome on 07 October. Discussion at the kickoff meeting indicated that some adaptation to our planned efforts would be helpful. This was neither surprising nor difficult to accommodate, but was a point at which we needed to make changes from our initial plan. The principal changes were (1) to redirect UW-P away from efforts to explore machine learning for file type detection; and (2) to increase the planned number of plug-ins of each type so as to provide greater insight into the feasibility of the plug-in architecture. The plan is now to emphasize the use of multiple plug-ins (both for file type detection and steganalysis) to exercise the plug-in architectural concept, giving insight into feasibility of the approach. UW-P refocused onto creation of plug-ins implementing well-known steganalysis algorithms.

UW-P delivered (to ECC) initial test versions of file type detectors for JPEG and GIF image file types that use file name extensions and magic bytes to type files. They also collected about 1G of file examples "from the wild" for testing, and set up four separate steganography programs to prepare "known bad" versions.

ECC implemented a controller that

- checks a startup directory for plug-ins
- loads any plug-ins detected and calls their init() function
- monitors specified directories for file creation
- copies files created in the specified directories to a work/quarantine area
- calls the loaded plug-ins against the files in the work area
- returns a "good/bad" call for each file

The controller does not currently distinguish between file type detection and steganalysis plug-ins. It therefore does not yet attempt to restrict calls to steganalysis plug-ins to files of an appropriate type.

We did encountered a technical issue in the plug-in API. Our initial plan was to have the controller component open each suspect file, then pass a handle to that open file to each plug-in as an argument to the typeof() function. This would minimize file manipulation overhead by amortizing the cost of a single file open/close over all invoked plug-ins. Unfortunately, it has proved to be quite complex to accomplish if the controller and plug-in are written in different languages, since each language has a language-specific representation of an open file. So, for example, C++ has a representation that differs

from Java, and it is not clear how to pass the C++ representation to a Java plug-in. Fortunately, this is not a critical issue for feasibility - premature optimization can be the enemy of good design. For the present, we are passing file names (with path) to the plug-ins, and require the plug-ins to do file open/close as needed.

November 2004

UW-P created initial test versions of file type detectors for JPEG and GIF image file types using file name extensions and magic bytes to type files. As these plug-ins were made available to the controller, we found some instances where the initial plug-in API definition was deficient, so it was updated. File type detection is now accomplished through a newly defined function `isOfType()`, analogous to `detect()` for steganalysis plug-ins. We clarified input and output parameter types and legal values and now require that the `detect()` function return a floating point value between 0.0 and 1.0 inclusive rather than a Boolean to accommodate statistical responses. Finally, we eliminated the `typeof()` function (which had provided a subset of the functionality required of `init()`).

The controller was enhanced to distinguish between file type detection and steganalysis plug-ins, and to apply only relevant analysis techniques based on the computed file type. UW-P updated their existing plug-ins to reflect the revised API definition.

December 2004

Most of our effort during December focused on completing the basic controller functionality. While this was primarily a matter of resolving issues surfaced by testing, we also added a simple user interface offering the following functions:

- specifying directories to monitor for new file creation
- starting and stopping the file creation monitoring thread
- turning logging on and off
- forcing an immediate scan of a single file or all files in a specific directory
- setting the threshold at which a file is considered "suspicious" (based on return values from steganalysis plug-ins)
- viewing a list of suspicious files

The controller currently does the following:

- checks a startup directory for plug-ins, loads any plug-ins detected and calls their init() function
- monitors any specified directories for file creation
- copies files created in the specified directories to a work/quarantine area
- calls the loaded plug-ins against the files in the work area
- checks the value returned from the steganalysis plug-ins against a user-defined threshold to determine if the file is "suspicious"
- clears all non-suspicious files from the work area

- reports suspicious files to the user
- (optionally) logs all activity

Most of the issues we have encountered in building the controller are system-level:

- detecting file creation/modification/deletion involves using an operating system service (true for both Windows and Unix)
- runtime discovery and invocation to plug-ins uses operating system calls and insight into parameter layout and passing conventions
- isolating file change detection from file analysis (to avoid causing the analysis process to introduce a bottleneck) is a threading/process issue
- detecting and recovering from plug-in failure also requires isolation through thread/process architecture

The current controller uses a threading model for isolation. We are finding the thread monitoring capabilities of our development environment to be somewhat limited, and are considering moving from threading to separate processes in some cases.

The plug-in API has proven robust so far, but has not had extensive testing with analysis plug-ins. We have discovered situations in which it would be helpful to have file type detectors return a numeric assessment rather than just a file type - for example, we have found situations in which image files lack "magic bytes" that the file type definition indicates they should contain. Technically, such files are not valid (e.g., are, strictly speaking, not really JPEG files), but are properly displayed by the tools we have employed, and so should be checked by the relevant analysis tools.

While we do not intend to devote any Phase I development effort to it, we have begun to give some thought to possible future work. One topic of discussion has been support for server-side use. The modularization of the system architecture appears to make it straightforward to retarget from our current client-side service orientation by simply replacing the file change detection component (and, of course, the user interface). On the server, the StegKit controller could be used as a module in a mail server (filtering emails before forwarding to individual clients), or as a filtering application within a proxy cache like Squid. By attaching to a cache, it would be possible to share the steganalysis facility without having to install on every user machine. A second area of concern is remote administration, allowing a centralized system administration group to configure and manage StegKit clients on end-user computers. We hope to develop some use cases to capture our discussions in these areas.

January 2005

With a working controller in place, efforts in January focused on plug-in development and testing. New steganalysis and file type detection plug-ins were provided by UWP, and proved useful in validating the operation of the controller (as well as performing their basic functions). The analysis plug-ins were developed by taking existing steganalysis

code (typically found on the Internet) and wrapping it to meet the plug-in API required by the controller.

This approach drew our attention back to issues of security. One of the plug-ins had an advertised flaw when used against valid images of a particular type (24 bit BMP)[1]. However, it crashed when presented with images of a slightly different type (16 bit BMP). The wrapping process did not attempt to correct the error, so when a test file invoked the plug-in on a 16 bit BMP, it crashed. In the initial controller implementation, this crash propagated back to the controller and caused it to crash in turn.

We have also been reimplementing the controller functionality in Java for greater portability, so the crash isolation issue will be addressed in the update. Our Java implementation does rely heavily on JNI functionality to access operating system services, such as file creation notification.

We now have examples of plug-ins written in both C++ and C.

UWP has had a paper accepted for presentation at the 38th Annual Midwest Instruction and Computing Symposium, April 8-9, 2005.

February 2005

We discovered situations in which it would be helpful to have file type detectors return a numeric assessment rather than just a file type - for example, we have found situations in which image files lack "magic bytes" that the file type definition indicates they should contain. Technically, such files are not valid (e.g., are, strictly speaking, not really JPEG files), but are properly displayed by the tools we have employed, and so should be checked by the relevant analysis tools.

Reimplementation of the controller functionality in Java for greater portability continues. We are using an inexpensive ($95) commercial development library, JNIWrapper, to reduce development time. JNIWrapper has no runtime license fees, so redistribution of StegKit has no impediments.

The project wiki is moribund - the last meaningful update was in November 2004. In talking with project members, it appears that the work has been effectively partitioned between ECC and UW-P, and that participants at each site find it easier to simply "walk down the hall" than to use the wiki. There is a fair amount of email traffic that probably could have been managed through the wiki, but we would have to consider the wiki experiment a failure to date.

We have created fewer steganalysis plug-ins than we had planned. Our approach has been to use existing open-source components, recast as plug-ins. Unfortunately, the code that

[1] "Note: This test will not work on 24-bit color images, because S-Tools hides the secret message a different way. However, with 24-bit images, a program is not required because these images are so rare their use would itself be suspicious"

we have found is filled with old software idioms that are proving difficult to reverse or translate. We anticipate that the production rate for steganalysis plug-ins will increase considerably in the next quarter.

March 2005

We completed the Java reimplementation of the StegKit controller. We also added new plug-ins for analysis of GIF files.

We prepared for and hosted at ECC on 24 March an interim review with participants from AFRL and speakerphone connection to UW-P. A beta version of StegKit (including a complete Windows install package and full source directories) was released to AFRL.

The interim review was most helpful in improving our understanding of the overall setting in which this project is placed. In particular, AFRL emphasized the value of being able to support steganalysis algorithm implementations that are not implemented as DLLs, and that may not match our current (and admittedly simple) model of returned information.

Based on feedback received at the interim review, we are revisiting the StegKit architecture to ensure ability to support already-developed steganalysis components available to AFRL, and to reflect our experience in creating the prototype.

Also discussed were other potential features and enhancements to the existing system, including (but not limited to):

- a more graceful response to changes in the threshold setting that would result in already analyzed files no longer exceeding the threshold - at a minimum, the user should be warned of the consequences.
- techniques for fusing data from multiple steganalysis plug-ins. This fusion could take place on several levels. In the past, we have discussed a "voting" approach to determining the likelihood of steganographic content, but other possibilities include getting detection information from one plug-in and content characteristics from another.
- richer response actions, such as selective alarms based on user role.
- performance issues, including the potential of executing plug-ins in individual threads.

April 2005

Based on feedback received at the interim review (see March status report), we designed and tested an approach for supporting non-DLL plug-ins.

The basic issue is that the StegKit controller architecture to this point has presumed that plug-ins will be implemented as DLLs (or, on Unix-like systems, as shared object files) exposing a specified API. While this works for the case where plug-ins are developed

with knowledge of the controller, it is less satisfactory for cases where there is an existing codebase developed without consideration of the specific controller/plug-in architecture. The design case was motivated by a projected commercialization of StegKit as a utility package analogous to anti-virus software - in that setting, the expectation of plug-ins developed to match a defined API is reasonable, even if the plug-ins are developed by a third party. However, an equally valid design case would be that described by AFRL, where there is a requirement to use existing steganalysis tool implementations without modifying the tested, working code to make it fit the specified StegKit DLL/API model.

There are basically four issues that must be handled: (1) discovery; (2) invocation protocol; (3) API mapping; and (4) additional (unanticipated) functionality.

Discovery: In the previous StegKit architecture, plug-ins were auto-discovered by searching a specified directory for DLLs. Coupled with monitoring of file creation events, this also makes it possible to dynamically incorporate new plug-ins into the overall functionality without stopping and restarting the controller. In the revised architecture, there must be a way to know where the plug-in functionality is provided.

Invocation protocol: When plug-ins are DLLs, they can be dynamically loaded and registered for use. The plug-in definition need not be available at the controller compile time. The COM-based DLL support in Windows (and similar shared object support in Unix) provides a means for the controller to find and invoke specific functions within a DLL. The StegKit plug-in API definition constrains the DLL functions so that there is no uncertainty regarding the invocation protocol. A similar capability must be provided for command line executables (which may take a variety of parameters, sometimes in flexible order) and Java jar files. In the case of Java jars, a strong presumption of the Java programming model is not satisfied, namely, that the jar file will be available at controller compilation time.

API mapping: The StegKit API is very simple. There are three functions, and any plug-in is required to provide two of them.[2] However, pre-existing candidate plug-ins, such as the stegdetect command line program, do not supply these functions directly. There must be a method for supplying a "virtual API" function from the capability of the target plug-in.

Additional (unanticipated) functionality: The StegKit API is also relatively impoverished in terms of what it expects from plug-ins. In particular, the controller tags files under consideration with a float between 0.0 and 1.0, intended to represent the likelihood that the file contains steganographic content. Many steganalysis algorithms, and the code that implements them, are capable of providing much more information. Stegdetect, for example, offers a guess as to what steganographic program was used to insert the surreptitious content. Other algorithms are capable of indicating the location of the hidden content, or even the detected content itself. There should be a way for these algorithms to supply this additional information to users, even though the existing StegKit model does not anticipate it.

[2] All plug-ins are required to supply an init() function. Plug-ins also supply either an isOfType() function or a detect() function depending on whether the plug-in is a file type detector or steganalyzer.

Fortunately, we have been able to generalize the StegKit architecture to accommodate a much wider range of plug-in components. Our specific new targets are self-contained executables that can be invoked from a command line (e.g., stegdetect, Under The Carpet) and Java class/jar files. We anticipate that the same techniques that we are developing can be used for Matlab programs, but have not confirmed this.

In the revised architecture, there is only slight change to the controller module itself. It continues to manage the process of detecting new file creation and application of file detection and steganalysis functions implemented as plug-ins. Plug-ins are no longer constrained to be DLLs implementing a specific API. Any impedance mismatch between the controller and the actual plug-ins is handled by a set of proxies or shims that act as transducers.

Instead of scanning for DLLs and loading them, on startup, the controller looks for what we're calling configuration files, each of which describes a single plug-in. A configuration file addresses each of the four issues discussed above. Based on the configuration file, the controller selects a proxy (e.g., a Java jar proxy), initializes it to handle interaction with the actual plug-in component, then treats the proxy as though it is the plug-in, making defined API calls and getting responses that are API-compliant.

The proxy exposes an API-compliant interface to the controller and maps the controller's function calls into invocations of the actual plug-in in accordance with the directions in the configuration file.

We have implemented demonstration cases for command line executables (stegdetect) and for Java jar files (locally written). Command line executables are invoked through a command interpreter, and Java jar functionality is accessed through reflection.

We have also revisited the literature describing existing steganalysis algorithms with the intent of better defining the content of configuration files. For each algorithm we are characterizing (to the extent that the available descriptions permit) the input parameters, output data, and target file type (e.g., JPEG, MP3).

May 2005

As a part of our literature review of existing steganalysis algorithms, we contacted some algorithm developers to determine the implementation language used for the software reported in their papers.

We prepared a version of StegKit that incorporated Java jar, Windows DLL, and command line plug-in launch directly from the controller, by incorporating the previously developed proxy code into the controller. A final distribution was prepared with both an installable Java-based StegKit and full source for the controller, team-developed plug-ins, and stegdetect (the third-party application that was used as a command line interface plug-in).

Results Obtained

In Phase I work we were successful in supporting Microsoft DLLs through the COM interface, a command line EXE through a proxy wrapper, and a Java application through reflection. It should be mentioned that each of these execution environments was engineered specifically for the individual application. This route is not suitable for a production framework and we have begun to analyze more flexible options.

Building the Phase I Framework to support steganalysis provided many learning experiences for our team. Our results include:

- The design and implementation of a Controller based architecture that supports multiple file type and steganalysis plug-ins through a simple API.
- Research, design and development of file type detector plug-ins that can be used to efficiently target a file with an appropriate steganalysis program. If an inappropriate steganalysis tool is used on a file we have observed undesirable behaviors including: crashes and the return of false positive or miss staganalysis results. The file type detection methods implemented or studied in Phase I include:
  - simple file type detection based on file extension (implemented),
  - file type detection based on magic bytes (implemented), and
  - file type detection based on signatures detection using the frequency of different byte values and identification of structures like color pallets, etc. (prototyped).
- Design and development of an architecture that supports the loading of an arbitrary number of file type and steganalysis plug-ins without recompiling the Controller. Plug-ins that conform to a simple API can be saved to a specific directly at any time. The Controller scans this directory at startup to maintain a current list of active plug-ins. This plug-in maintenance method could easily be extended to trigger on an interval or event basis.
- Design and development of a graphical user interface that permits an administrator to configure and control the Phase I framework. This control included such functionality as:
  - manual file scanning of a specified file system directory or individual file,
  - automatic background file scanning based on the operating system sending notification to the framework when files were created in a specified file system directory,
  - a simple "what-if" analysis tool that allowed filtering logged results to return the files that had specific estimated probability of steganographic content as determined by previous steganalysis.
- Investigation and implementation of a framework that includes the use of INI configuration files that define the interface between plug-ins and the Controller. The INI configuration files allow for a much wider variety of plug-in execution modes, inputs and outputs.

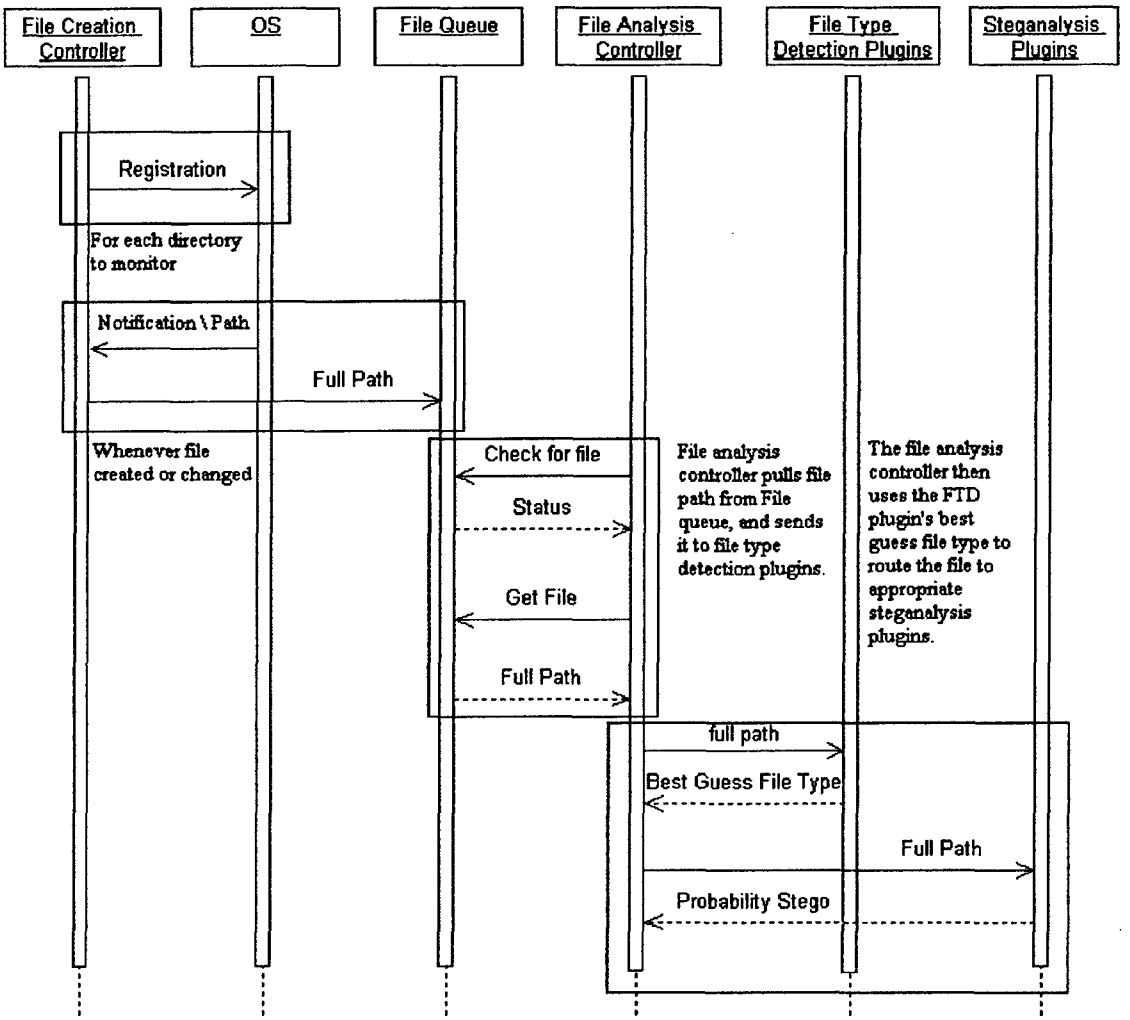
- Survey of the steganalysis literature and determination that a Phase II Framework will need to support multiple runtime environments in both command line (console) and GUI modes in order to integrate steganalysis from many different sources. The runtime environments include:
  - Java JVM
  - EXE
  - .NET
  - MATLAB®, and
  - Microsoft DLLs.
- Design and development of prototypes that support steganalysis plug-ins written in three different languages and run in three very different execution modes:
  - Microsoft DLL using COM to communicate input and output parameters,
  - Java using Java reflection, and
  - EXE command line programs using redirection of standard input and output

Below in Figure 1 is a high-level sequence diagram for the Phase I StegKit. The File Creation Controller and File Queue determine the existence of a new file and queue it for processing, respectively. The File Creation Controller is configured through the GUI to attend to any file activity in a particular part of the file system.

The Phase I controller has a great deal of knowledge of the overall StegKit process built into it. During its startup process, it searches the StegKit directory for plug-ins and loads and registers any that it finds. It initializes each plug-in, collecting information about the plug-in type and applicability. After initialization, the controller receives notification of new objects to be analyzed either from the user interface (manual requests to check a specific file or directory) or from the file creation monitor. It then runs all file type analysis plug-ins and, based on the results of that analysis, any relevant steganalysis plug-ins. Logging, notification, and quarantine operations are part of the controller.

We defined an API for plug-in components that would be invoked by a central controller. The Phase I API was very simple, requiring that (a) plug-ins be implemented as Windows Dynamic Link Libraries (DLLs), and (b) each plug-in supply two functions: init() and either isOfType() (for file type detection plug-ins) or detect() (for steganalysis plug-ins). The arguments and return values for the required functions were specified in the API. We demonstrated a working implementation of this architecture at the interim review held at ECC's facility.

Figure 1: Phase I controller sequence diagram



At that review, the Government representatives pointed out that it would be desirable if StegKit could accommodate previously created, or legacy steganalysis applications that were (a) not implemented as DLLs, and (b) did not support the defined StegKit API. Furthermore, these legacy steganalysis applications were implemented in a variety of technologies, including command line executables, Java class libraries (typically structured as jar files), and MATLAB® scripts, and could not be modified to match the StegKit plug-in implementation model or API. While this was somewhat different than what we had understood from the original topic description and our initial interactions, it is clearly a matter of interest that would greatly extend the utility of StegKit for users such as the Government technical advisors. Accordingly, we devoted a considerable amount of our remaining Phase I effort to investigating the issues.

We constructed proxies that wrapped the stegdetect command line executable program and an internally-prepared Java class, packaged in a jar file. The proxies were Windows DLLs offering the defined StegKit API. When an API function was called, the proxy

would respond appropriately, if necessary invoking functionality within its wrapped component and interpreting any response from that wrapped component. As an illustration, consider the stegdetect proxy.

- The stegdetect component has no functionality that corresponds to the StegKit init() API function, so all that functionality has to be provided by the proxy. The proxy simply responds with the plug-in type (steganalysis) and file type (JPG).
- For the detect() API function, the proxy receives the file path in the detect() call and uses it to build a stegdetect command line with the file path as the main argument, a parameter that controls output format (suppressing output when there is no detection), and output redirection to a known file name. The command is then executed in an inferior process, and the redirected output is examined for one or more asterisks enclosed in parentheses, which is how stegdetect rates the likelihood of steganographic content. The number of asterisks is mapped to a value between 0 and 1, which is then returned by the proxy as the value of the detect() call.

No change was needed to the controller - command line and Java jar plug-ins were accessible through proxies that satisfied the StegKit plug-in API while using the non-standard components.

As a proof-of-concept, our attempt to wrap non-compliant components was a success. However, the proxies/wrappers were hand-constructed, requiring programmer involvement to create. This would (probably) not be acceptable as a solution for the general legacy component problem - after all, if programming was an acceptable cost, then the legacy components could be modified to satisfy the StegKit API.

Estimate of Technical Feasibility

In Phase I, we demonstrated the feasibility of our general technical architecture featuring a centralized controller with plug-in components for file type detection and steganalysis. Placing overall control in a centralized component removes dependency on the specific details of any specific steganalysis algorithm, and, when coupled with file detection plug-ins, makes it possible to avoid unproductive computational expenditure that would be caused if steganalysis algorithms are run against suspect material for which they are inappropriate. We created an implementation of the architecture that used operating system services to detect file creation events, and then applied the type detection and analysis process to each changed file. By running as a service under the Windows operating system, the implementation was capable of monitoring all content encountered by a user as they browsed the Web.

In our Phase I proof-of-concept, plug-in components were required to provide a particular application programming interface (API), as suggested by the topic announcement. However, this requirement to comply with an API specification precluded the use of many existing implementations of steganalysis algorithms. First, since the existing implementations pre-dated the API specification, they would have to be revised to

comply with the specification. Second, and possibly more critical, the current API specification makes some strong assumptions regarding the nature of the analytical result, and these assumptions are not valid for all algorithms. Accordingly, practical considerations dictate that the StegKit architecture be enhanced to support "legacy" steganalysis implementations without requiring that they be rewritten.

Furthermore, because there are constant advances in steganography, StegKit must be capable of adapting to new approaches to data hiding, as well as incorporating improvements in steganalytical algorithms. The plug-in architecture enables the needed flexibility: new plug-ins may be provided to address new steganographic challenges, or to improve on existing algorithms. It is highly desirable that new and revised plug-ins be accessible without having to stop and restart the StegKit application - there should be no "reboot" required. We identified, but did not demonstrate in Phase I, an approach to "hot-plugging" that would detect and use new plug-ins without halting StegKit execution.

While the general feasibility of the approach has been established, there are still many issues to be addressed to produce a complete engineering prototype, suitable for deployment in an enterprise setting and ready for final product polish leading to commercial availability. We have identified the following as "next step" technical objectives:

1. Extend architecture to support "legacy" steganalysis algorithm implementations.
2. Support "hot-plug" extensibility.
3. Generalize and enrich the logging and reporting/notification capability.
4. Extend applicability by handling compressed file types (zip, gzip, etc.).
5. Support alternate file sources, such as mail servers, specifically Microsoft Exchange.
6. Improve the manageability of the system.

While not a specific technical objective, it would also be highly desirable to offer support for alternate operating system platforms, with Linux and Solaris as specific targets.

Cumulative List of Persons Involved

At ECC: Robert Bechtel (PI), Bill Turney
At UW-P: Mike Rowe (Co-PI), Daine Lesniak, Doug Hickok

Publications

Lesniak D, Hickok D, and Rowe M, "File Type Detection Technology", *38th Annual Midwest Instruction and Computing Symposium*, April 8-9, 2005.

StegKit Final Phase I Release Notes
31 May 2005

There is one CD in the Final release. It contains both a Windows-compatible install package and source code for the StegKit controller (both Java and Visual Basic) and plug-ins (both file type detection and steganalysis).

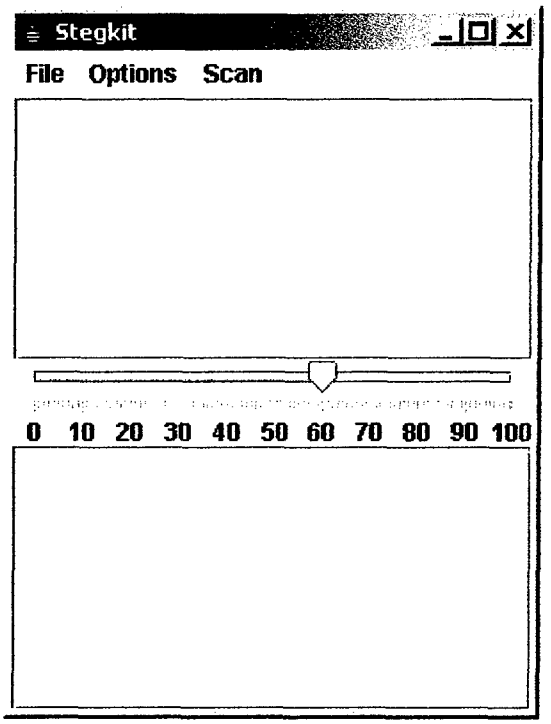
Installation:

Insert the Install CD in a CD drive. Run SETUP.EXE from the CD. You'll be asked for a target directory - the default is \Program Files\jStegKit. An entry will be added to the Start menu.

Operations:

Startup

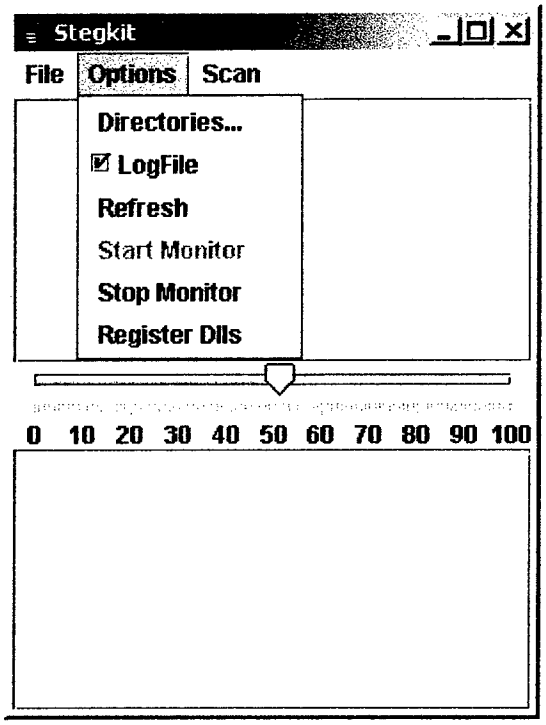
Select Start->Programs->jStegKit. This should launch the application, bringing up the main StegKit window, as shown below:



The slider between the windows sets the threshold at which an analyzed file will be quarantined.

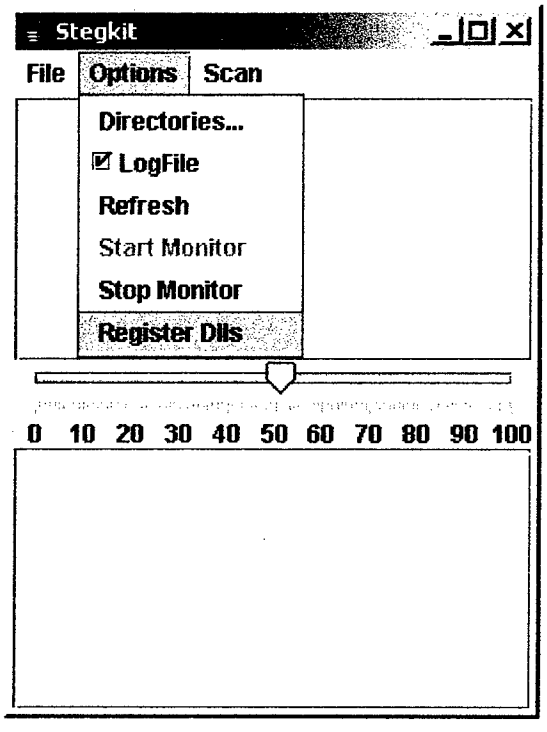
Enable/Disable Logging

Typical first actions include activating logging by selecting Options->LogFile. (This is a toggle - if selected when checked, logging will be disabled.)

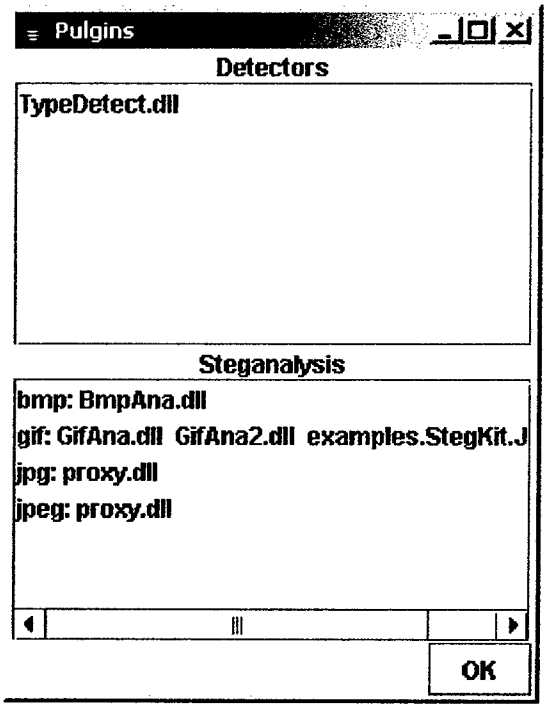


View Plug-Ins

You may select the "Register Dlls" option to see what plug-ins are currently available and loaded in StegKit.



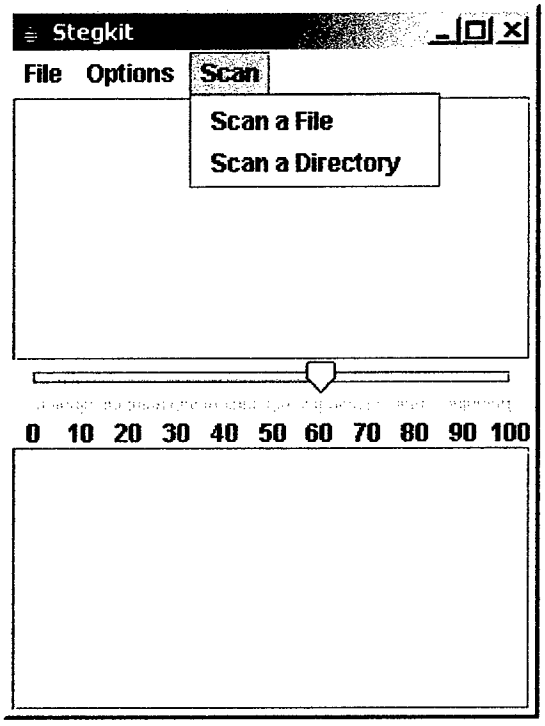
The plug-ins are displayed in a separate window, and are distinguished as to whether they are file type detectors or steganalysis components. Steganalysis components are further identified by the type of file they have identified as being useful against.



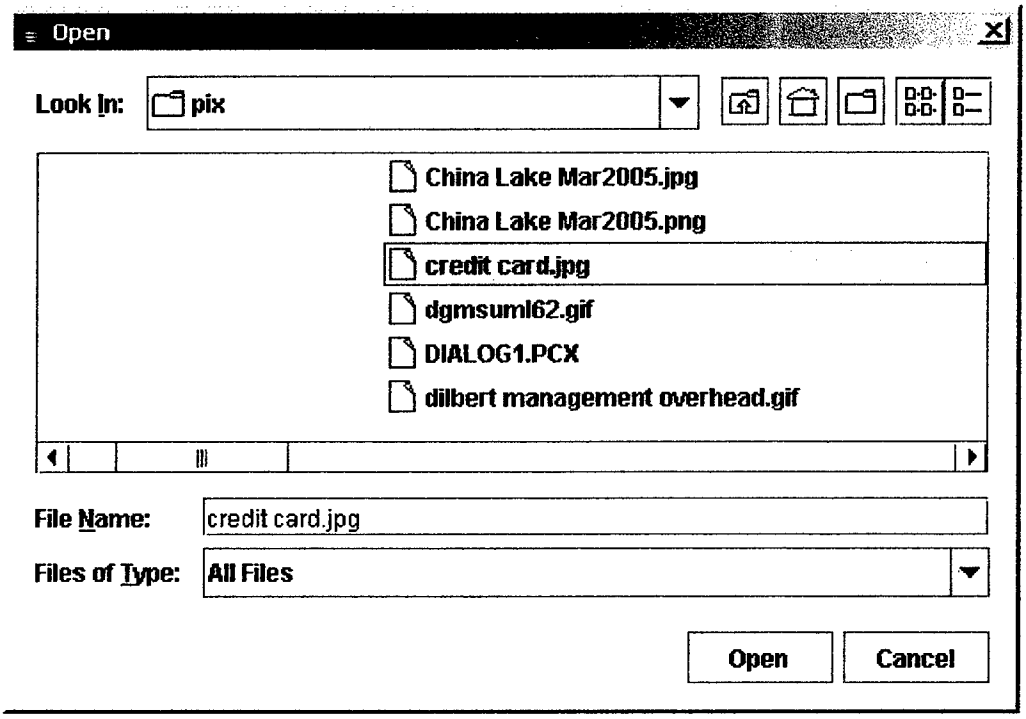
Click on the "OK" button to dismiss the plug-in listing.

Manually Scan Files or Directories

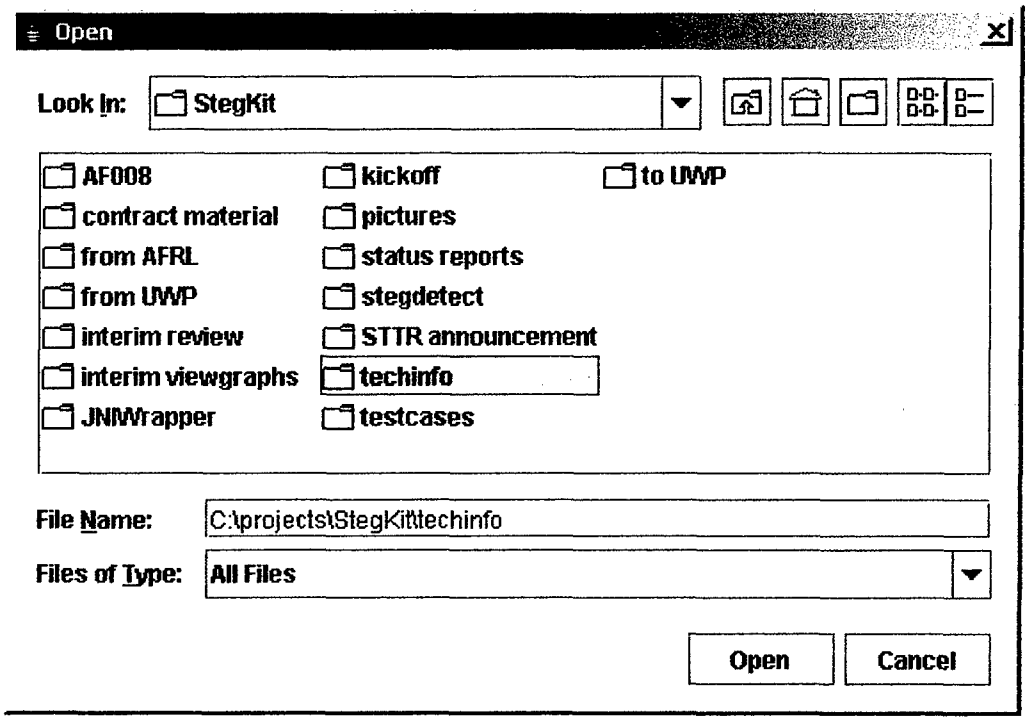
Individual files or specified directories may be scanned by using the Scan menu.



You will be offered a directory browser to select the desired file

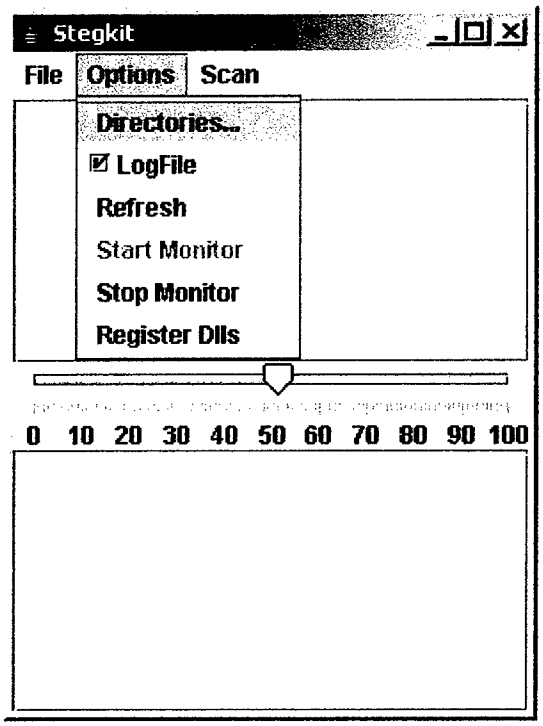


or directory, as appropriate.

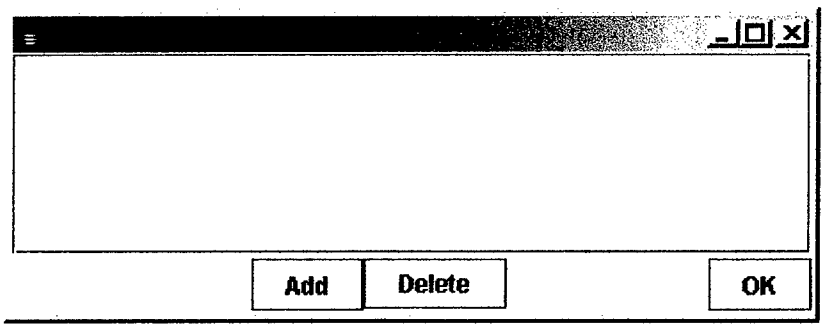


Monitoring Directories

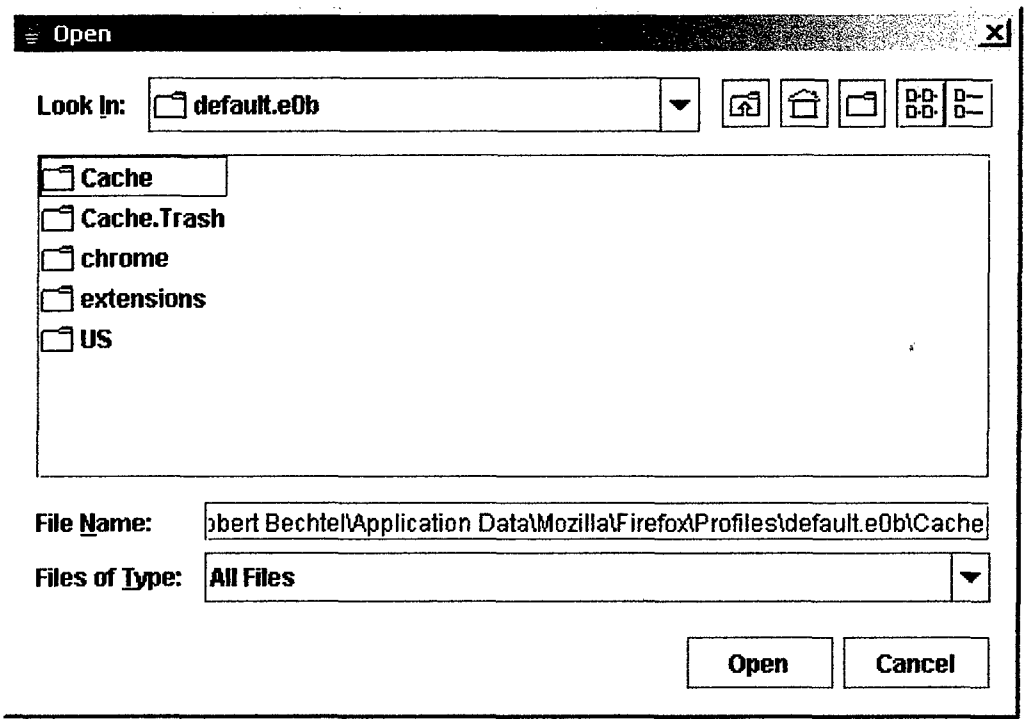
To monitor a directory, first select Options->Directories.



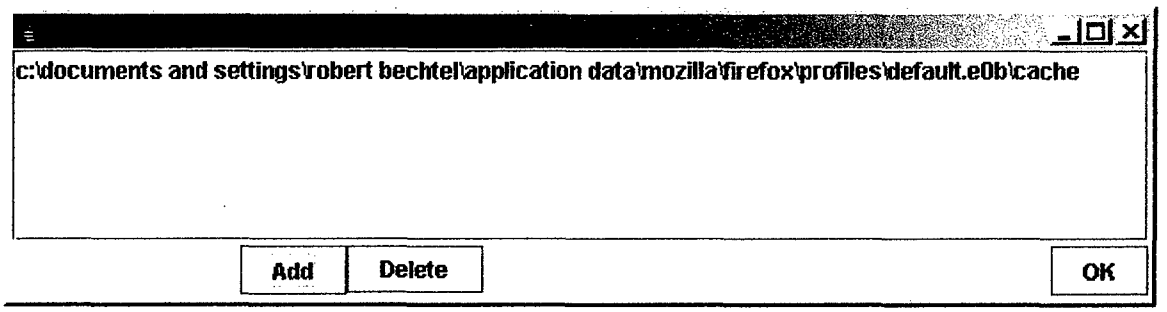
This will bring up a list of monitored directories (which should be empty on the first run).



Click on the Add button to bring up a directory browser, and browse to the directory to be monitored:



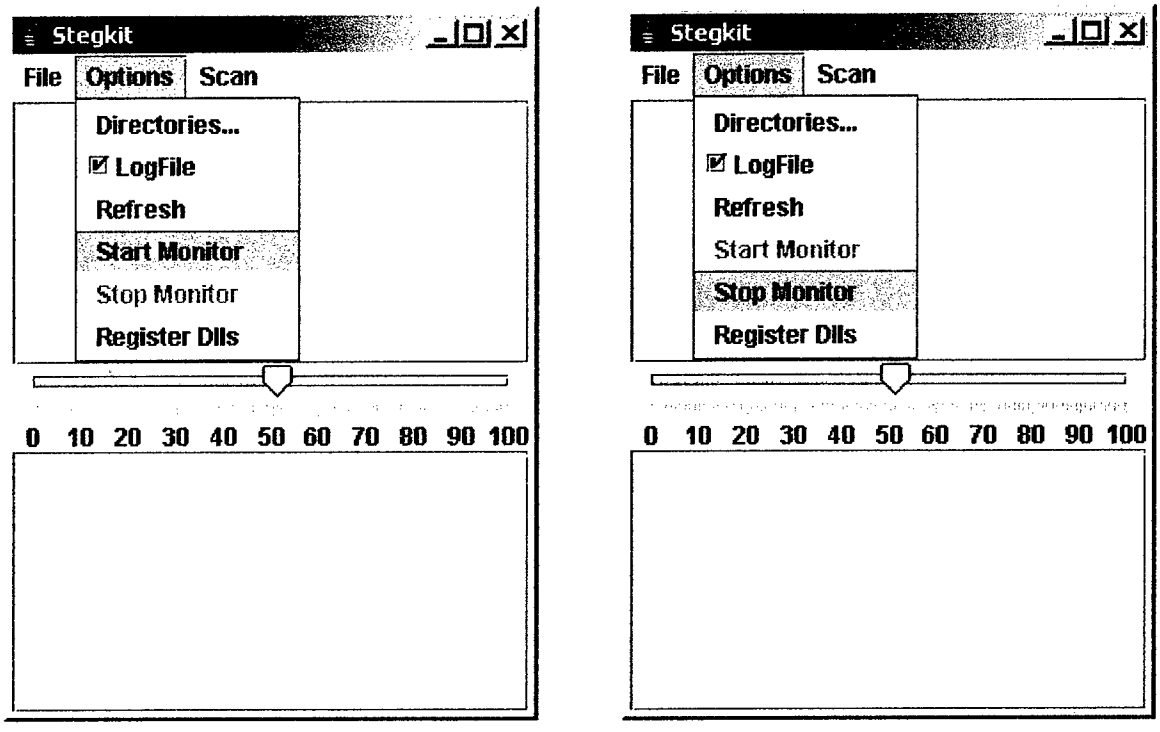
When the desired directory has been selected, click on Open to add the directory to the list of monitored directories.



Click on OK to return to the main StegKit window, or click on Add to monitor additional directories. If you want to remove a directory from monitoring, click on it in the directory list to select it, then click on the Delete button.

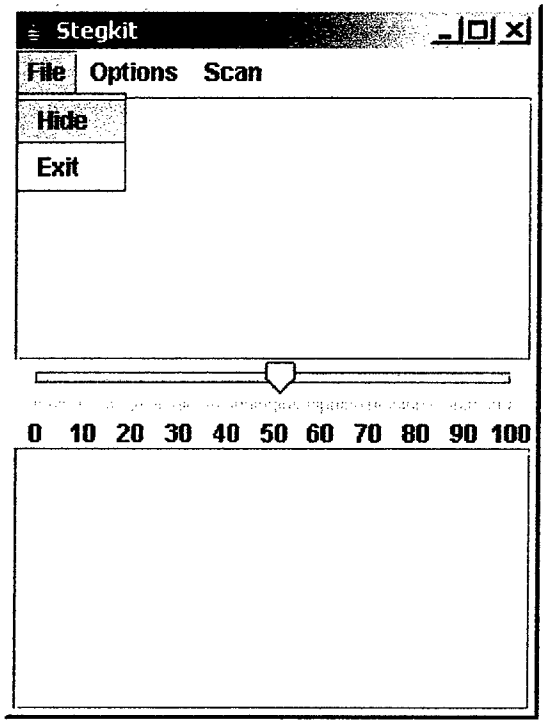
Activating and Deactivating Directory Monitoring

Monitoring can be turned off and on using the Options->Stop Monitor and Options->Start Monitor menu selections. Start Monitor will be inactive and gray when the monitor is running, and Stop Monitor will be inactive and gray when the monitor is stopped.



Hiding StegKit

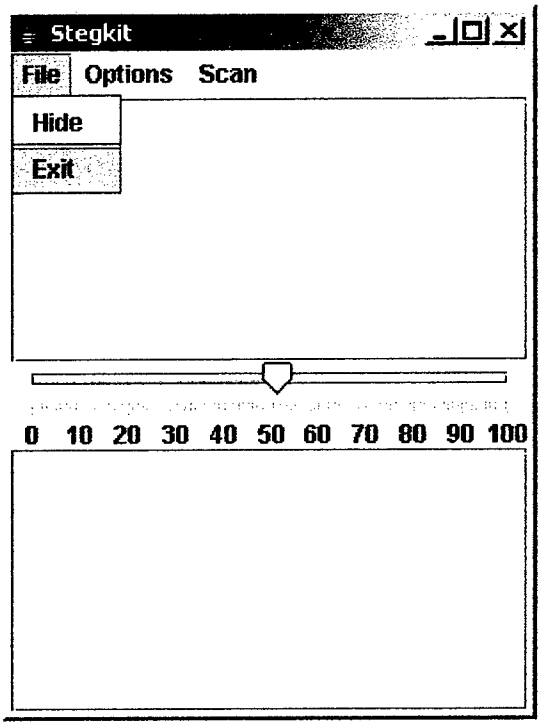
While running, the StegKit window may be hidden (leaving an icon on the system tray) by selecting File->Hide.



To restore, right click on the icon in the system tray and select Show StegKit from the popup menu.

Exiting from StegKit

When StegKit window is visible: Select File->Exit.



When the StegKit window is hidden, right click on the StegKit icon in the system tray, then select Exit from StegKit from the popup menu.

Log files and quarantine:

The StegKit log file is named logfile.txt and is stored in the directory where StegKit was installed (default: \Program Files\jStegKit). It is just a text file, and may be viewed with any program capable of displaying text files.

The StegKit install directory also holds a subdirectory named Quaratine. Copies of quarantined files (those that exceed the user-set threshold) are placed here. The subdirectory also includes a file named quarn.txt that includes a cross-reference of all quarantined files.

NOTE: The quarantine area is sensitive to changes in the threshold. Increasing the threshold can remove a file from the quarantine area, if the threshold moves above the "probable detection level" associated with the file.

Distribution CD:

The root directory contains a Windows install of JStegKit. Simply execute setup.exe to run the installer.

```
\ (CD top level)
   0x0409.ini
   instmsia.exe
   instmsiw.exe
   isscript.msi
   jStegKit.msi
   setup.exe
   Setup.ini
```

The program files directory contains the executable files for the proxy-based JStegKit, and includes a StegDetect distribution (both source and executable).

```
\ (CD top level)\program files
      \ (CD top level)\program files\jStegKit
         BmpAna.dll
         GifAna.dll
         GifAna2.dll
         JGifAna.class
         JGifAna.java
         jniwrap-2.7.1.jar
         jniwrap.dll
         jniwrap.lic
         proxy.dll
         Steg.ico
         StegFound.ico
         StegKit.cfg
         StegKit.exe
         StegKit.jar
         StegKit.jarContent
         StegKit.jfg
```

```
TypeDetect.dll
winpack.jar
   \ (CD top level)\program files\jStegKit\StegDetect
      cygwin1.dll
      gdk-1.3.dll
      glib-1.3.dll
      gmodule-1.3.dll
      gnu-intl.dll
      gtk-1.3.dll
      iconv-1.3.dll
      JPHide.jpg
      JSteg-Limpia.jpg
      JSteg-Sucia.jpg
      LEGAL.NOTICE
      looney.jpg
      nickel_a.jpg
      nickel_b.jpg
      OnTheRoad.jpg
      README
      README.txt
      rules.ini
      sd.exe
      stegbreak.exe
      stegbreak.pdf
      stegdetect-0.5.tar.gz
      stegdetect.exe
      stegdetect.pdf
      unicodebo.pdf
      xsteg.exe
```

The StegSource directory contains source code for a variety of pieces.

BmpAna - BMP steganalysis
GifAna - GIF steganalysis
GifAna2 - another GIF steganalyzer
jStegKit - jStegKit controller source
proxy - proxy for StegDetect
StegIntf - DLL interface for VB version of StegKit
TypeDetect - file type detector plugins
VBSteg - source for VB version of StegKit

```
\ (CD top level)\StegSource
      \ (CD top level)\StegSource\BmpAna
         BmpDect.cpp
         BmpDect.def
         BmpDect.dsp
         BmpDect.dsw
         BmpDect.h
         BmpDect.ncb
         BmpDect.opt
         BmpDect.plg
         BmpDetector.h
         ExeDetector.h
         FileDect2.cpp
```

```
   FileDect2.h
   FileDetectionControler.h
   FileDetectionUtilities.h
   FileTypeDetector.h
   GifDetector.h
   JpegDetector.h
   MovDetector.h
   Mp3Detector.h
   PngDetector.h
   ReadMe.txt
   StdAfx.cpp
   StdAfx.h
   WaveDetector.h
   ZipDetector.h
\ (CD top level)\StegSource\GifAna
   BmpDetector.h
   datatype.h
   endianio.c
   endianio.h
   ExeDetector.h
   FileDect2.cpp
   FileDect2.h
   FileDetectionControler.h
   FileDetectionUtilities.h
   FileTypeDetector.h
   gif.h
   GifDect.cpp
   GifDect.def
   GifDect.dsp
   GifDect.dsw
   GifDect.h
   GifDect.ncb
   GifDect.opt
   GifDect.plg
   GifDetector.h
   gifread.c
   JpegDetector.h
   MovDetector.h
   Mp3Detector.h
   PngDetector.h
   ReadMe.txt
   StdAfx.cpp
   StdAfx.h
   WaveDetector.h
   ZipDetector.h
\ (CD top level)\StegSource\GifAna2
   BmpDetector.h
   datatype.h
   endianio.c
   endianio.h
   ExeDetector.h
   FileDect2.cpp
   FileDect2.h
   FileDetectionControler.h
   FileDetectionUtilities.h
   FileTypeDetector.h
   gif.h
```

```
    GifDect.cpp
    GifDect.h
    GifDetect2.cpp
    GifDetect2.def
    GifDetect2.dsp
    GifDetect2.dsw
    GifDetect2.h
    GifDetect2.ncb
    GifDetect2.opt
    GifDetect2.plg
    GifDetector.h
    gifread.c
    JpegDetector.h
    MovDetector.h
    Mp3Detector.h
    PngDetector.h
    ReadMe.txt
    StdAfx.cpp
    StdAfx.h
    WaveDetector.h
    ZipDetector.h
\ (CD top level)\StegSource\jStegKit
    .nbattrs
    Directory$1.class
    Directory$2.class
    Directory$3.class
    Directory$4.class
    Directory.class
    Directory.form
    Directory.java
    DllPlugIn.class
    DllPlugIn.java
    JavaPlugIn.class
    JavaPlugIn.java
    JGifAna.class
    JGifAna.java
    PluginList$1.class
    PluginList$2.class
    PluginList.class
    PluginList.form
    PluginList.java
    Steg.ico
    StegFound.ico
    StegKit$1.class
    StegKit$10.class
    StegKit$11.class
    StegKit$12.class
    StegKit$13.class
    StegKit$14.class
    StegKit$15.class
    StegKit$16.class
    StegKit$17.class
    StegKit$18.class
    StegKit$19.class
    StegKit$2.class
    StegKit$20.class
    StegKit$21.class
```

```
   StegKit$22.class
   StegKit$23.class
   StegKit$24.class
   StegKit$3.class
   StegKit$4.class
   StegKit$5.class
   StegKit$6.class
   StegKit$7.class
   StegKit$8.class
   StegKit$9.class
   StegKit$GetNewFile$CSecurityAttributes.class
   StegKit$GetNewFile.class
   StegKit$NotifyIconData.class
   StegKit$TrayIcon$TrayIconWindowProc.class
   StegKit$TrayIcon.class
   StegKit$TrayIconSample.class
   StegKit.cfg
   StegKit.class
   StegKit.form
   StegKit.java
   StegPlugIns.class
   StegPlugIns.java
   TestPlugIn.class
   TestPlugIn.java
\ (CD top level)\StegSource\proxy
   proxy.cpp
   proxy.def
   proxy.dsp
   proxy.dsw
   proxy.h
   proxy.ncb
   proxy.opt
   proxy.plg
   ReadMe.txt
   StdAfx.cpp
   StdAfx.h
\ (CD top level)\StegSource\StegIntf
   ReadMe.txt
   StdAfx.cpp
   StdAfx.h
   StegIntf.cpp
   StegIntf.def
   StegIntf.dsp
   StegIntf.dsw
   StegIntf.h
   StegIntf.ncb
   StegIntf.opt
   StegIntf.plg
\ (CD top level)\StegSource\TypeDetect
   BmpDetector.cpp
   BmpDetector.h
   ExeDetector.cpp
   ExeDetector.h
   FileDect2.cpp
   FileDect2.def
   FileDect2.dsp
   FileDect2.dsw
```

```
   FileDect2.h
   FileDect2.ncb
   FileDect2.opt
   FileDect2.plg
   FileDetectionControler.cpp
   FileDetectionControler.h
   FileDetectionUtilities.cpp
   FileDetectionUtilities.h
   FileTypeDetector.cpp
   FileTypeDetector.h
   GifDetector.cpp
   GifDetector.h
   JpegDetector.cpp
   JpegDetector.h
   MovDetector.cpp
   MovDetector.h
   Mp3Detector.cpp
   Mp3Detector.h
   PngDetector.cpp
   PngDetector.h
   ReadMe.txt
   StdAfx.cpp
   StdAfx.h
   TestMain.cpp
   WaveDetector.cpp
   WaveDetector.h
   ZipDetector.cpp
   ZipDetector.h
\ (CD top level)\StegSource\VBSteg
   frmDirectories.frm
   frmoutput.frm
   frmoutput.frx
   Module1.bas
   Project1.vbp
   Project1.vbw
```

Appendix B - Source Code Listings

C++ code (file type detection plug-ins, proxy code)

BmpDect.h, .cpp, .def
BmpDetector.h, .cpp
ExeDetector.h, .cpp
FileDect2.h, .cpp, .def
FileDetectionControler.h, .cpp
FileDetectionUtilities.h, .cpp
FileTypeDetector.h, .cpp
GifDect.h, .cpp, .def
GifDetect2.h, .cpp, .def
GifDetector.h, .cpp
JpegDetector.h, .cpp
MovDetector.h, .cpp
Mp3Detector.h, .cpp
PngDetector.h, .cpp
proxy.h, .cpp, .def
StegIntf.h, .cpp, .def
WaveDetector.h, .cpp
ZipDetector.h, .cpp

Java code (StegKit controller, one test plug-in)

Directory.java, .form
DllPlugIn.java
JavaPlugIn.java
JGifAna.java
PluginList.java, .form
StegKit.java, .form
StegPlugIns.java
TestPlugIn.java

```
#ifndef _BMPDECT_H
#define _BMPDECT_H

// The following ifdef block is the standard way of creating macros which make exporting
// from a DLL simpler. All files within this DLL are compiled with the FILEDECT2_EXPORTS
// symbol defined on the command line. this symbol should not be defined on any project
// that uses this DLL. This way any other project whose source files include this file see
// FILEDECT2_API functions as being imported from a DLL, wheras this DLL sees symbols
// defined with this macro as being exported.
#ifdef BMPDECT_EXPORTS
#define BMPDECT_API __declspec(dllexport)
#else
#define BMPDECT_API __declspec(dllimport)
#endif


#ifdef __cplusplus
extern "C" {
#endif

BMPDECT_API int _stdcall init(
    char valids[10]
    );

BMPDECT_API float _stdcall detect(
    const char* file_name
    );


#ifdef __cplusplus
}
#endif
#endif
```

```
// FileDect2.cpp : Defines the entry point for the DLL application.
//
#include "stdafx.h"
#include "BmpDect.h"
#include "FileDetectionControler.h"

// added by Doug --------
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <memory.h>

typedef struct _bmp_header{
    char ID[2];
    int size;
    char res1[2], res2[2];
    int offset;
} bmp_header;

typedef struct _bmp_info{
    int size, width, height;
    int planes, BitsPerPixel;
    int compress, BMsize;
    int Hres, Vres;
    int ColorsUsed, ColorsImportant;
} bmp_info;
// ---------------------------------


BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}

// DLL init function
BMPDECT_API int _stdcall init(
    char valids[10]
    )
{
    strcpy(valids,"BMP");
    return 1;
}

// Added by Doug (rest of file)
/* prototypes */
void load_bmp_header( FILE *BMP, bmp_header * BH);
void load_bmp_info(FILE *BMP, bmp_info *BI);

// DLL detect function
BMPDECT_API float _stdcall detect(
    const char* file_name
    )
{
    // I'm doing some heavy editing to make a command line program
    // from a book into a plug-in, so hopefully it goes ok.
    int i, j, i1;
    int BMPsize, num_colors, dup_color;
    FILE *BMPin;
    unsigned char color_map[1024];
    bmp_header BMPHead;
    bmp_info BMPInfo;
```

```
    BMPin = fopen( file_name, "rb" );
        if( BMPin ==0) {
            //fprintf( stderr, "unable to open input file\n" );
            //exit(1);
            return .5; //might have stego, might not -- cant open the file
        }

        fseek( BMPin, 0, 2);
        BMPsize = ftell(BMPin);
        fseek(BMPin, 0, 0 );

        load_bmp_header( BMPin, &BMPHead );
        load_bmp_info( BMPin, &BMPInfo );

        /* calculate the number of colors in the pallet */
        num_colors = (BMPHead.offset - 14 - BMPInfo.size)/4;

        fread( color_map, 1, BMPInfo.ColorsUsed*4, BMPin);

        dup_color = 0;

        // check if it's too big to handle...  (Doug)
        // "too big" means it has 16 million colors or more
        if (BMPInfo.ColorsUsed < 16000000) {

            /*find duplicate colors */
            for( i=0; i< BMPInfo.ColorsUsed; i++) {
                for( j=0; j<BMPInfo.ColorsUsed; j++) {
                    i1= abs(color_map[4*i] - color_map[4*j] ) +
                        abs(color_map[4*i+1] - color_map[4*j+1]) +
                        abs(color_map[4*i+2] - color_map[4*j+2]);
                    if((i1<4) && (i != j)) {dup_color++; }
                }
            }

        } //end of Doug's IF block

        fseek(BMPin, BMPHead.offset, 0);

        //printf( "duplicate colors: ", dup_color );

        //if( dup_color < 100 ) printf( "data had not been hidden with stools" );
        //if( dup_color >= 100 ) printf( "data has been hidden with stools");
        //printf( "\n\n" );

        fclose(BMPin);

        if( dup_color >= 100 ) return .95;
        return .05;
}

// private function to load the BMP header info
void load_bmp_header( FILE *BMP, bmp_header *BH) {
    unsigned char buffer[16];

    memset(BH, 0, sizeof(bmp_header));
    fread(buffer, 1, 14, BMP);

    memcpy(BH->ID, buffer, 2);

    memcpy(&(BH->size), buffer+2, 4);
    memcpy(BH->res1, buffer+6, 2);
    memcpy(BH->res2, buffer+8, 2);
    memcpy(&(BH->offset), buffer+10, 4);
}

// private function to load the BMP's info
void load_bmp_info( FILE *BMP, bmp_info *BI){
    unsigned char buffer[112];

    memset( BI, 0, sizeof(bmp_info));
    fread(buffer, 1, 108, BMP);
```

```
    memcpy(&(BI->size), buffer, 4);
    // *** Yes, we're ignoring some errors for now... ***
    //if((BI->size != 40) && (BI->size != 108)) {
    //  printf( "bitmap info header is size %d - cannot handle this format\n", BI->size);
    //  exit(1);
    //}

    /*reset postion to beginning of color table */
    if( BI->size == 40) {fseek(BMP, -68, 1);}
    memcpy( &(BI->width), buffer+4, 4);
    memcpy( &(BI->height), buffer+8, 4);
    BI->planes = buffer[12]+256*buffer[13];
    BI->BitsPerPixel = buffer[14]+256*buffer[15];

    memcpy(&(BI->compress), buffer+16, 4);
    memcpy(&(BI->BMsize), buffer+20, 4);
    memcpy(&(BI->Hres), buffer+24, 4);
    memcpy(&(BI->Vres), buffer+28, 4);
    memcpy(&(BI->ColorsUsed), buffer+32, 4);
    memcpy(&(BI->ColorsImportant), buffer+36, 4);

    if(BI->ColorsUsed == 0 ){
        BI->ColorsUsed = 1<< BI->BitsPerPixel;
    }
}
```

```
LIBRARY BmpDect
EXPORTS
    init
    detect
```

```
#ifndef BmpDetector_h
#define BmpDetector_h

#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
using namespace std;

//-------------------------------------------------------------------------------------
//this class is a child of FileTypeDetector that is used to determine if a file is a
//BMP or not.
//-------------------------------------------------------------------------------------
class BmpDetector : public FileTypeDetector
{
    private:

        int header_length;
        unsigned char* header_magic_bytes;

        FileDetectionUtilities* utilities;

    public:
        //-------------------------------------------------------------------------------
        //returns true if the file indicated by file_name is a BMP and false otherwise
        //-------------------------------------------------------------------------------
        virtual bool isOfType( const char* file_name );

        //-------------------------------------------------------------------------------
        //contructor for BmpDetector, takes a pointer to the FileDetectionUtilities
        //instance it is to utilize
        //-------------------------------------------------------------------------------
        BmpDetector( FileDetectionUtilities* pointer_to_utilites );
};

#endif
```

```
#include "BmpDetector.h"
#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"

using namespace std;

BmpDetector::BmpDetector( FileDetectionUtilities* pointer_to_utilities )
{
    header_length = 2;
    utilities = pointer_to_utilities;
    header_magic_bytes = new unsigned char[header_length];


    //initialize header_magic_bytes to reflect the magic bytes in a bmp header
    header_magic_bytes[0] = 0x42;
    header_magic_bytes[1] = 0x4d;
}

bool BmpDetector::isOfType( const char* file_name )
{
    bool is_bmp = true;
    if( !utilities->matchesHeader( file_name, header_magic_bytes, header_length, 0 ) )
        is_bmp = false;

    return is_bmp;
}
```

```
#ifndef ExeDetector_h
#define ExeDetector_h

#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
using namespace std;

//-------------------------------------------------------------------------------------
//this class is a child of FileTypeDetector that is used to determine if a file is a
//Exe or not.
//-------------------------------------------------------------------------------------
class ExeDetector : public FileTypeDetector
{
    private:
        int header_length;
        unsigned char* header_magic_bytes;
        FileDetectionUtilities* utilities;

    public:
        //---------------------------------------------------------------------------
        //returns true if the file indicated by file_name is a Exe and false otherwise
        //---------------------------------------------------------------------------
        virtual bool isOfType( const char* file_name );

        //---------------------------------------------------------------------------
        //contructor for ExeDetector, takes a pointer to the FileDetectionUtilities
        //instance it is to utilize
        //---------------------------------------------------------------------------
        ExeDetector( FileDetectionUtilities* pointer_to_utilites );
};

#endif
```

```
#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
#include "ExeDetector.h"

using namespace std;

ExeDetector::ExeDetector( FileDetectionUtilities* pointer_to_utilities )
{
    header_length = 2;
    utilities = pointer_to_utilities;
    header_magic_bytes = new unsigned char[header_length];


    //initialize header_magic_bytes to reflect the magic bytes in a Exe header
    header_magic_bytes[0] = 0x4d;
    header_magic_bytes[1] = 0x5a;
}

bool ExeDetector::isOfType( const char* file_name )
{
    bool is_exe = true;
    if( !utilities->matchesHeader( file_name, header_magic_bytes, header_length, 0 ) )
        is_exe = false;
    return is_exe;
}
```

```
#ifndef _FILEDECT2_H
#define _FILEDECT2_H

// The following ifdef block is the standard way of creating macros which make exporting
// from a DLL simpler. All files within this DLL are compiled with the FILEDECT2_EXPORTS
// symbol defined on the command line. this symbol should not be defined on any project
// that uses this DLL. This way any other project whose source files include this file see
// FILEDECT2_API functions as being imported from a DLL, wheras this DLL sees symbols
// defined with this macro as being exported.
#ifdef FILEDECT2_EXPORTS
#define FILEDECT2_API __declspec(dllexport)
#else
#define FILEDECT2_API __declspec(dllimport)
#endif


#ifdef __cplusplus
extern "C" {
#endif

FILEDECT2_API int _stdcall init(
    char *valids
    );

FILEDECT2_API char* _stdcall isOfType(
    const char *file_name
    );


#ifdef __cplusplus
}
#endif
#endif
```

```
// FileDect2.cpp : Defines the entry point for the DLL application.
//
#include <stdio.h>
#include "stdafx.h"
#include "FileDect2.h"
#include "FileDetectionControler.h"

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}

FILEDECT2_API int _stdcall init(
    char *valids
    )
{
    strcpy(valids,"");
    return 0;
}



FILEDECT2_API char* _stdcall isOfType(
    const char *file_name
    )

{
    char *extension_list;

    extension_list = (char *)malloc( 256 );


    FileDetectionControler* my_controler = new FileDetectionControler();

    strcpy(extension_list,my_controler->doesExtensionMatchFile( file_name ));


    return extension_list;
}
```

```
LIBRARY FileDect2
EXPORTS
    init
    isOfType
```

```
#ifndef FileDetectionControler_h
#define FileDetectionControler_h

#include "FileTypeDetector.h"
#include "JpegDetector.h"
#include "GifDetector.h"
#include "BmpDetector.h"
#include "WaveDetector.h"
#include "PngDetector.h"
#include "MovDetector.h"
#include "ZipDetector.h"
#include "ExeDetector.h"
#include "Mp3Detector.h"
#include <map>
#include <string>
#include "FileDetectionUtilities.h"

using namespace std;

//-------------------------------------------------------------------------------------
//This class controls filetypedetection, it is used to determine if a filetype matches
//its extension.
//-------------------------------------------------------------------------------------
class FileDetectionControler
{
    private:
        map< string, FileTypeDetector*> extension_detector_map;
        FileDetectionUtilities util;

    public:
        //-------------------------------------------------------------------------------
        //contructor for FileDetectionControler, will throw an exception when passed
        //an extension it doesnt recognize. It recognizes jpg, jpeg, gif, png, bmp, wav,
        //exe, mov, zip, and mp3
        //-------------------------------------------------------------------------------
        FileDetectionControler();

        //-------------------------------------------------------------------------------
        //returns true if file indicated by file_name matches its extension and false
        //otherwise.
        //-------------------------------------------------------------------------------
        char* doesExtensionMatchFile( const char* file_name );

};




#endif
```

```
#include "FileDetectionControler.h"
#include "FileTypeDetector.h"
#include "JpegDetector.h"
#include "GifDetector.h"
#include "BmpDetector.h"
#include "WaveDetector.h"
#include "PngDetector.h"
#include "MovDetector.h"
#include "ZipDetector.h"
#include "ExeDetector.h"
#include "Mp3Detector.h"
#include "FileDetectionUtilities.h"
#include <map>
#include <string>

using namespace std;

FileDetectionControler::FileDetectionControler()
{
    util = FileDetectionUtilities();
    JpegDetector* my_jpeg_detector = new JpegDetector( &util );
    GifDetector* my_gif_detector = new GifDetector( &util );
    BmpDetector* my_bmp_detector = new BmpDetector( &util );
    WaveDetector* my_wave_detector = new WaveDetector( &util );
    PngDetector* my_png_detector = new PngDetector( &util );
    MovDetector* my_mov_detector = new MovDetector( &util );
    ZipDetector* my_zip_detector = new ZipDetector( &util );
    ExeDetector* my_exe_detector = new ExeDetector( &util );
    Mp3Detector* my_mp3_detector = new Mp3Detector( &util );
    extension_detector_map["jpeg"] = my_jpeg_detector;
    extension_detector_map["jpg"] = my_jpeg_detector;
    extension_detector_map["gif"] = my_gif_detector;
    extension_detector_map["bmp"] = my_bmp_detector;
    extension_detector_map["wav"] = my_wave_detector;
    extension_detector_map["png"] = my_png_detector;
    extension_detector_map["mov"] = my_mov_detector;
    extension_detector_map["zip"] = my_zip_detector;
    extension_detector_map["exe"] = my_exe_detector;
    extension_detector_map["mp3"] = my_mp3_detector;
}

char* FileDetectionControler::doesExtensionMatchFile( const char* file_name )
{
    char extension[200];
    int file_name_index = 0;
    char temp_char = file_name[ file_name_index ];
    int length;

    extension[0] = '\0';
    while( temp_char != '\0' )
    {
        temp_char = tolower( temp_char );
        length = strlen(extension);
        extension[length] = temp_char;
        extension[length + 1] = '\0';
        if( temp_char == '.' )
            extension[0] = '\0';
        file_name_index++;
        temp_char = file_name[ file_name_index ];
    }

    //need to add stuff for not being able to find it
    FileTypeDetector* detector_to_use = extension_detector_map[extension];
    if (detector_to_use == NULL)
    {
        strcpy(extension,"");
        return "";
    }
    if (detector_to_use->isOfType( file_name ) == true)
    {
        return extension;
    }
    else
```

```
    {
        strcpy(extension,"");
        return "";
    }
}
```

```
#ifndef FileDetectionUtilities_h
#define FileDetectionUtilities_h

#include <iostream>
#include <fstream>
#include <string>
using namespace std;

//-------------------------------------------------------------------------------------
//this class provides a collection of utility methods that are usefull for verifying \
//determining file types
//-------------------------------------------------------------------------------------
class FileDetectionUtilities
{

    public:

        //-----------------------------------------------------------------------------
        //determines if the magic bytes in the header match the magic bytes specified
        //by magic_bytes, num_magic_bytes indicates how many magic bytes there are, and
        //offset is the position that the header of the file starts with respect to the
        //beginning of the file.
        //-----------------------------------------------------------------------------
        bool matchesHeader( const char* file_name, const unsigned char* magic_bytes,
                                          int num_magic_bytes, int offset );
        //-----------------------------------------------------------------------------
        //determines if the magic bytes in teh trailer match the magic bytes specified
        //by magic_bytes. the trailer is considered to be the last num_bytes bytes of the
        //file
        //-----------------------------------------------------------------------------
        bool matchesTrailer( const char* file_name, const unsigned char* magic_bytes,
                             int num_bytes );

        //bool containsString( string file_name, string string_to_check_for );

        //int instancesOfString( string file_name, string string_to_count );

        //histogramer to be defined and added at a later date
};

#endif
```

```
#include "FileDetectionUtilities.h"
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

bool FileDetectionUtilities::matchesHeader( const char* file_name, const unsigned char* magic_byte
s,
                                            int num_magic_bytes, int offset )
{
    bool matches_header = true;
    unsigned char char_to_test;
    ifstream fin( file_name, ios::binary );
    fin.seekg( offset ,ios::beg);
    //throw exception if file isnt opened?
    for( int i = 0; i < num_magic_bytes; i++ )
    {
        fin >> char_to_test;
        if( !(magic_bytes[i] == char_to_test ) )
            matches_header = false;
    }
    fin.close();
    return matches_header;
}

bool FileDetectionUtilities::matchesTrailer( const char* file_name, const unsigned char* magic_byt
es,
                              int num_bytes )
{
    bool matches_trailer = true;
    unsigned char char_to_test;
    ifstream fin( file_name, ios::binary );
    int backup_index = -1 * num_bytes;
    fin.seekg( backup_index ,ios::end);
    for( int i = 0; i < num_bytes; i++ )
    {
        fin >> char_to_test;
        if( !(magic_bytes[i] == char_to_test ) )
            matches_trailer = false;
    }
    fin.close();
    return matches_trailer;
}
```

```
#ifndef FileTypeDetector_h
#define FileTypeDetector_h

#include <string>
#include "FileDetectionUtilities.h"
using namespace std;

//-----------------------------------------------------------------------------------------
//This class is the base FileTypeDetector from which detectors for specific file
//types are to be derived
//-----------------------------------------------------------------------------------------
class FileTypeDetector
{
    public:

        //-----------------------------------------------------------------------------
        //returns true, should be overidded by child classes to indicate whether the
        //file indicated by file_name is the file_type they specialize in verifying
        //-----------------------------------------------------------------------------
        virtual bool isOfType( const char* file_name );

};

#endif
```

```
#include <string>
#include "FileTypeDetector.h"
using namespace std;

 bool FileTypeDetector::isOfType( const char* file_name )
{
    return true;
}
```

```
#ifndef _GIFDECT_H
#define _GIFDECT_H

// The following ifdef block is the standard way of creating macros which make exporting
// from a DLL simpler. All files within this DLL are compiled with the DETECT2_EXPORTS
// symbol defined on the command line. this symbol should not be defined on any project
// that uses this DLL. This way any other project whose source files include this file see
// DETECT2_API functions as being imported from a DLL, wheras this DLL sees symbols
// defined with this macro as being exported.
#ifdef GIFDECT_EXPORTS
#define GIFDECT_API __declspec(dllexport)
#else
#define GIFDECT_API __declspec(dllimport)
#endif



#ifdef __cplusplus
extern "C" {
#endif

GIFDECT_API int _stdcall init(
    char valids[10]
    );

GIFDECT_API float _stdcall detect(
    const char *file_name
    );

#ifdef __cplusplus
}
#endif
#endif
```

```
// GifDect.cpp : Defines the entry point for the DLL application.
//

#include "stdafx.h"
#include "FileDetectionControler.h"
#include "GifDect.h"

//added by Doug
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <io.h>
#include "datatype.h"
#include "endianio.h"
#include "gif.h"


BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}



GIFDECT_API int _stdcall init(
    char* valids
    )
{
    strcpy(valids,"GIF");
    return 1;
}



GIFDECT_API float _stdcall detect(
    const char *file_name
    )

{
    register WORD i;        /* Loop counter                                    */
    FILE *fpGif;            /* Pointer to the input FILE stream                */
    GIFHEAD           gifHead;          /* GIF Header structure              */

    fpGif = fopen(file_name, "rb");

    /* Read the GIF image file header information */
    ReadGifHeader(&gifHead, fpGif);

    /*
    ** check for pattern madelsteg leaves if -fp option is not used
    */
    if ( (gifHead.PackedField & 0x80) && ((1L << ((gifHead.PackedField & 0x07) + 1) == 256 ) ) )
    {
        for (i = 0; i < 128; i++)
        {
            if( gifHead.GlobalCT[i].Red != gifHead.GlobalCT[ i + 128 ].Red ||
                gifHead.GlobalCT[i].Green != gifHead.GlobalCT[ i + 128 ].Green ||
                gifHead.GlobalCT[i].Blue != gifHead.GlobalCT[ i + 128 ].Blue )
                    return .05f;
        }
        return .95f;
```

```
    }
    return .05f;
}
```

```
LIBRARY GifDect
EXPORTS
    init
    detect
```

```
#ifndef _GIFDETECT2_H
#define _GIFDETECT2_H

// The following ifdef block is the standard way of creating macros which make exporting
// from a DLL simpler. All files within this DLL are compiled with the FILEDECT2_EXPORTS
// symbol defined on the command line. this symbol should not be defined on any project
// that uses this DLL. This way any other project whose source files include this file see
// FILEDECT2_API functions as being imported from a DLL, wheras this DLL sees symbols
// defined with this macro as being exported.
#ifdef GIFDETECT2_EXPORTS
#define GIFDETECT2_API __declspec(dllexport)
#else
#define GIFDETECT2_API __declspec(dllimport)
#endif


#ifdef __cplusplus
extern "C" {
#endif

GIFDETECT2_API int _stdcall init(
    char valids[10]
    );

GIFDETECT2_API float _stdcall detect(
    const char* file_name
    );

#ifdef __cplusplus
}
#endif
#endif
```

```
// FileDect2.cpp : Defines the entry point for the DLL application.
//
#include "stdafx.h"
#include "GifDetect2.h"
#include "FileDetectionControler.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <io.h>
#include "datatype.h"
#include "endianio.h"
#include "gif.h"

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}

GIFDETECT2_API int _stdcall init(
    char valids[10]
    )
{
    strcpy(valids,"GIF");
    return 1;
}



GIFDETECT2_API float _stdcall detect(
    const char *file_name
    )

{
    int i;
    int j;
    int similarity;
    FILE *fpGif;
    GIFHEAD           gifHead;
    long duplicate_colors = 0;
    int color_table_entries;

    fpGif = fopen(file_name, "rb");

    /* Read the GIF image file header information */
    ReadGifHeader(&gifHead, fpGif);

    if( gifHead.PackedField & 0x80 )
    {
        color_table_entries = (1L << ((gifHead.PackedField & 0x07) + 1));
        for( i = 0; i < color_table_entries; i++ )
        {
            for( j = 0; j < color_table_entries; j++ )
            {
                similarity = abs( gifHead.GlobalCT[i].Blue - gifHead.GlobalCT[j].Blue ) +
                    abs( gifHead.GlobalCT[i].Green - gifHead.GlobalCT[j].Green ) +
                    abs( gifHead.GlobalCT[i].Red - gifHead.GlobalCT[j].Red );
                if( (similarity < 4 ) && ( i != j ) )
                    duplicate_colors++;
            }
        }
```

```
        if( duplicate_colors >= 1000 )
            return .80;
        if( duplicate_colors >= 500 )
            return .25;
        if( duplicate_colors >= 100 )
            return .15;
    }
    return .05;
}
```

```
LIBRARY GifDect
EXPORTS
    init
    detect
```

```
#ifndef GifDetector_h
#define GifDetector_h

#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
using namespace std;

//-------------------------------------------------------------------------------------
//this class is a child of FileTypeDetector that is used to determine if a file is a
//gif or not.
//-------------------------------------------------------------------------------------
class GifDetector : public FileTypeDetector
{
    private:
        int shared_header_length;
        int trailer_length;
        unsigned char* header_magic_bytes_old;
        unsigned char* header_magic_bytes_new;
        unsigned char* trailer_magic_bytes;

        FileDetectionUtilities* utilities;

    public:
        //-------------------------------------------------------------------------
        //returns true if the file indicated by file_name is a gif and false otherwise
        //-------------------------------------------------------------------------
        virtual bool isOfType( const char* file_name );

        //-------------------------------------------------------------------------
        //contructor for GifDetector, takes a pointer to the FileDetectionUtilities
        //instance it is to utilize
        //-------------------------------------------------------------------------
        GifDetector( FileDetectionUtilities* pointer_to_utilites );
};




#endif
```

```
#include "GifDetector.h"
#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"

using namespace std;

GifDetector::GifDetector( FileDetectionUtilities* pointer_to_utilities )
{
    shared_header_length = 6;
    trailer_length = 2;
    utilities = pointer_to_utilities;
    header_magic_bytes_old = new unsigned char[shared_header_length];
    header_magic_bytes_new = new unsigned char[shared_header_length];

    trailer_magic_bytes = new unsigned char[trailer_length];


    //initialize header_magic_bytes_old to reflect the magic bytes in the old gif header
    header_magic_bytes_old[0] = 0x47;
    header_magic_bytes_old[1] = 0x49;
    header_magic_bytes_old[2] = 0x46;
    header_magic_bytes_old[3] = 0x38;
    header_magic_bytes_old[4] = 0x37;
    header_magic_bytes_old[5] = 0x61;

    //initialize header_magic_bytes_new to reflect the magic bytes in the new gif header
    header_magic_bytes_new[0] = 0x47;
    header_magic_bytes_new[1] = 0x49;
    header_magic_bytes_new[2] = 0x46;
    header_magic_bytes_new[3] = 0x38;
    header_magic_bytes_new[4] = 0x39;
    header_magic_bytes_new[5] = 0x61;


    //initialize trailer_magic_bytes to reflect the magic bytes in a gif
    trailer_magic_bytes[0] = 0x00;
    trailer_magic_bytes[1] = 0x3b;
}

bool GifDetector::isOfType( const char* file_name )
{
    bool is_gif = true;

    if( !( ( utilities->matchesHeader( file_name, header_magic_bytes_new,
                                       shared_header_length, 0) ) ||
           ( utilities->matchesHeader( file_name, header_magic_bytes_old,
                                       shared_header_length, 0) ) ) )
        is_gif = false;

    if( !utilities->matchesTrailer( file_name, trailer_magic_bytes, trailer_length ) )
        is_gif = false;

    return is_gif;
}
```

```
#ifndef JpegDetector_h
#define JpegDetector_h

#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
using namespace std;

//-------------------------------------------------------------------------------------------
//this class is a child of FileTypeDetector that is used to determine if a file is a
//jpeg or not.
//-------------------------------------------------------------------------------------------
class JpegDetector : public FileTypeDetector
{
    private:

        int header_length;
        unsigned char* header_magic_bytes;

        FileDetectionUtilities* utilities;

    public:
        //-------------------------------------------------------------------------------
        //returns true if the file indicated by file_name is a jpeg and false otherwise
        //-------------------------------------------------------------------------------
        virtual bool isOfType( const char* file_name );

        //-------------------------------------------------------------------------------
        //contructor for JpegDetector, takes a pointer to the FileDetectionUtilities
        //instance it is to utilize
        //-------------------------------------------------------------------------------
        JpegDetector( FileDetectionUtilities* pointer_to_utilites );
};

#endif
```

```
#include "JpegDetector.h"
#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"

using namespace std;

JpegDetector::JpegDetector( FileDetectionUtilities* pointer_to_utilities )
{
    header_length = 3;
    utilities = pointer_to_utilities;
    header_magic_bytes = new unsigned char[header_length];

    //initialize header_magic_bytes to reflect the magic bytes in a jpeg header
    header_magic_bytes[0] = 0xff;
    header_magic_bytes[1] = 0xd8;
    header_magic_bytes[2] = 0xff;
}

bool JpegDetector::isOfType( const char* file_name )
{
    bool is_jpeg = true;
    if( !utilities->matchesHeader( file_name, header_magic_bytes, header_length, 0 ) )
        is_jpeg = false;

    return is_jpeg;
}
```

```
#ifndef MovDetector_h
#define MovDetector_h


#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
using namespace std;

//-------------------------------------------------------------------------------------------
//this class is a child of FileTypeDetector that is used to determine if a file is a
//Mov or not.
//-------------------------------------------------------------------------------------------
class MovDetector : public FileTypeDetector
{
    private:
        int header_length;
        int header_offset;
        unsigned char* header_magic_bytes;
        FileDetectionUtilities* utilities;

    public:
        //---------------------------------------------------------------------------
        //returns true if the file indicated by file_name is a Mov and false otherwise
        //---------------------------------------------------------------------------
        virtual bool isOfType( const char* file_name );

        //---------------------------------------------------------------------------
        //contructor for MovDetector, takes a pointer to the FileDetectionUtilities
        //instance it is to utilize
        //---------------------------------------------------------------------------
        MovDetector( FileDetectionUtilities* pointer_to_utilites );
};

#endif
```

```
#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
#include "MovDetector.h"

using namespace std;

MovDetector::MovDetector( FileDetectionUtilities* pointer_to_utilities )
{
    header_length = 4;
    header_offset = 4;
    utilities = pointer_to_utilities;
    header_magic_bytes = new unsigned char[header_length];


    //initialize header_magic_bytes to reflect the magic bytes in a Mov header
    header_magic_bytes[0] = 0x6d;
    header_magic_bytes[1] = 0x6f;
    header_magic_bytes[2] = 0x6f;
    header_magic_bytes[3] = 0x76;

}

bool MovDetector::isOfType( const char* file_name )
{
    bool is_mov = true;
    if( !utilities->matchesHeader( file_name, header_magic_bytes, header_length, header_offset ) )
        is_mov = false;
    return is_mov;
}
```

```
#ifndef Mp3Detector_h
#define Mp3Detector_h


//#include "MagicByteMask.h"
#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
using namespace std;

//-----------------------------------------------------------------------------------------
//this class is a child of FileTypeDetector that is used to determine if a file is a
//Mp3 or not.
//-----------------------------------------------------------------------------------------
class Mp3Detector : public FileTypeDetector
{
    private:
        int possible_header_length_a;
        int possible_header_length_b;
        unsigned char* possible_header_magic_bytes_a;
        unsigned char* possible_header_magic_bytes_b;
        FileDetectionUtilities* utilities;

    public:
        //-------------------------------------------------------------------------------
        //returns true if the file indicated by file_name is a Mp3 and false otherwise
        //-------------------------------------------------------------------------------
        virtual bool isOfType( const char* file_name );

        //-------------------------------------------------------------------------------
        //contructor for Mp3Detector, takes a pointer to the FileDetectionUtilities
        //instance it is to utilize
        //-------------------------------------------------------------------------------
        Mp3Detector( FileDetectionUtilities* pointer_to_utilites );
};

#endif
```

```
#include "Mp3Detector.h"
#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"

using namespace std;

Mp3Detector::Mp3Detector( FileDetectionUtilities* pointer_to_utilities )
{
    possible_header_length_a = 2;
    possible_header_length_b = 3;
    utilities = pointer_to_utilities;
    possible_header_magic_bytes_a = new unsigned char[possible_header_length_a];
    possible_header_magic_bytes_b = new unsigned char[possible_header_length_b];

    possible_header_magic_bytes_a[0] = 0xff;
    possible_header_magic_bytes_a[1] = 0xe0; // thru FF

    possible_header_magic_bytes_b[0] = 0x49;
    possible_header_magic_bytes_b[1] = 0x44;
    possible_header_magic_bytes_b[2] = 0x33;
}

bool Mp3Detector::isOfType( const char* file_name )
{
    bool is_mp3 = true;
    bool is_in_mp3_range = false;
    unsigned char* temp_header_magic_bytes = new unsigned char[possible_header_length_a];

    if( !( utilities->matchesHeader( file_name, possible_header_magic_bytes_b,
                                     possible_header_length_b, 0) ) )
        is_mp3 = false;

    temp_header_magic_bytes[0] = possible_header_magic_bytes_a[0];
    for( int i = 0; i < 32; i++ )
    {
        temp_header_magic_bytes[1] = possible_header_magic_bytes_a[1] + i;
        if( utilities->matchesHeader( file_name, temp_header_magic_bytes,
                                      possible_header_length_a, 0) )
            is_in_mp3_range = true;
    }

    is_mp3 = is_mp3 || is_in_mp3_range;
    return is_mp3;
}
```

```
#ifndef PngDetector_h
#define PngDetector_h


#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
using namespace std;

//-------------------------------------------------------------------------------------------
//this class is a child of FileTypeDetector that is used to determine if a file is a
//Png or not.
//-------------------------------------------------------------------------------------------
class PngDetector : public FileTypeDetector
{
    private:
        int header_length;
        unsigned char* header_magic_bytes;
        FileDetectionUtilities* utilities;

    public:
        //---------------------------------------------------------------------------------
        //returns true if the file indicated by file_name is a Png and false otherwise
        //---------------------------------------------------------------------------------
        virtual bool isOfType( const char* file_name );

        //---------------------------------------------------------------------------------
        //contructor for PngDetector, takes a pointer to the FileDetectionUtilities
        //instance it is to utilize
        //---------------------------------------------------------------------------------
        PngDetector( FileDetectionUtilities* pointer_to_utilites );
};

#endif
```

```
#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
#include "PngDetector.h"

using namespace std;

PngDetector::PngDetector( FileDetectionUtilities* pointer_to_utilities )
{
    header_length = 4;
    utilities = pointer_to_utilities;
    header_magic_bytes = new unsigned char[header_length];


    //initialize header_magic_bytes to reflect the magic bytes in a Png header
    header_magic_bytes[0] = 0x89;
    header_magic_bytes[1] = 0x50;
    header_magic_bytes[2] = 0x4e;
    header_magic_bytes[3] = 0x47;
}

bool PngDetector::isOfType( const char* file_name )
{
    bool is_png = true;
    if( !utilities->matchesHeader( file_name, header_magic_bytes, header_length, 0 ) )
        is_png = false;
    return is_png;
}
```

```
#ifndef _PROXY_H
#define _PROXY_H

#ifdef PROXY_EXPORTS
#define PROXY_API __declspec(dllexport)
#else
#define PROXY_API __declspec(dllimport)
#endif



#ifdef __cplusplus
extern "C" {
#endif

PROXY_API int _stdcall init(
    char valids[10]
    );

PROXY_API float _stdcall detect(
    const char* file_name
    );


#ifdef __cplusplus
}
#endif
#endif
```

```
// proxy.cpp : Defines the entry point for the DLL application.
//

#include "stdafx.h"
#include "stdlib.h"
#include <stdio.h>
#include "proxy.h"

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}


// DLL init function
PROXY_API int _stdcall init(
    char valids[10]
    )
{
    strcpy(valids,"JPG:JPEG");
    return 1;
}

PROXY_API float _stdcall detect(
    const char* file_name
    )
{
    char StCommand[256];
    char line[256];
    FILE *pFile;
    float ans;


    strcpy(StCommand,"stegdetect\\stegdetect -q ");
    //strcpy(StCommand,"java JAna ");
    strcat(StCommand,file_name);
    strcat(StCommand," > bbb.out");
    system(StCommand);
    pFile = fopen ("bbb.out","rt");
    if ( fgets(line, 256, pFile) == NULL)
    {
        ans = 0.0;
    }
    else
    {
        if (strstr(line,"(***)") != NULL) {
            ans = 1.0;
        }
        else if (strstr(line,"(**)") != NULL) {
            ans = 0.67;
        }
        else if (strstr(line,"(*)") != NULL) {
            ans = 0.33;
        }
        else {
            ans = 0.0;
        }
    }


    fclose (pFile);
    return ans;
```

```
}
```

```
LIBRARY proxy
EXPORTS
    init
    detect
```

```
#ifndef _STEGINTF_H
#define _STEGINTF_H

// The following ifdef block is the standard way of creating macros which make exporting
// from a DLL simpler. All files within this DLL are compiled with the STEGINTF_EXPORTS
// symbol defined on the command line. this symbol should not be defined on any project
// that uses this DLL. This way any other project whose source files include this file see
// STEGINTF_API functions as being imported from a DLL, wheras this DLL sees symbols
// defined with this macro as being exported.
#ifdef STEGINTF_EXPORTS
#define STEGINTF_API __declspec(dllexport)
#else
#define STEGINTF_API __declspec(dllimport)
#endif


#ifdef __cplusplus
extern "C" {
#endif

#define  MAX_MSG_LEN  255    // max length of a returned Msg[] array.


STEGINTF_API int _stdcall InitCall(
    HINSTANCE hinstLib,
    char *RetString
   );

STEGINTF_API int _stdcall IsTypeCall(
    //char FileName[MAX_MSG_LEN]
    HINSTANCE hinstLib,
    char *FileName,
    char *RetString
   );

STEGINTF_API float _stdcall DetectCall(
    HINSTANCE hinstLib,
    char *FileName
   );


#ifdef __cplusplus
}
#endif
#endif
```

```
// StegIntf.cpp : Defines the entry point for the DLL application.
//

#include "stdafx.h"
#include "StegIntf.h"

#ifdef __cplusplus
extern "C" {
#endif

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}


typedef int (CALLBACK* INITOF)(char*);


STEGINTF_API int _stdcall InitCall(
    HINSTANCE hinstLib,
    char *RetString
    )
{
    INITOF ProcAdd;
    int StegType;

    ProcAdd = (INITOF) GetProcAddress(hinstLib, "init");
    if (NULL == ProcAdd)
        return -1;
    StegType = (ProcAdd) (RetString);
    return (StegType);
};



//typedef int (*ISTYPEOF)(char*);

typedef char* (CALLBACK* ISTYPEOF)(LPSTR);

STEGINTF_API int _stdcall IsTypeCall(
    //char FileName[MAX_MSG_LEN]
    HINSTANCE hinstLib,
    char *FileName,
    char *RetString
    )
{
    ISTYPEOF ProcAdd;
    char *pRet;

    ProcAdd = (ISTYPEOF) GetProcAddress(hinstLib, "isOfType");
    if (NULL == ProcAdd)
        return -1;
    pRet = ProcAdd (FileName);
    strcpy(RetString,pRet);

    return (1);
};

typedef float (CALLBACK* DECTECTOF)(LPSTR);

STEGINTF_API float _stdcall DetectCall(
```

```
    HINSTANCE hinstLib,
    char *FileName
   )
{
    DECTECTOF ProcAdd;
    float pRet;

    ProcAdd = (DECTECTOF) GetProcAddress(hinstLib, "detect");
    if (NULL == ProcAdd)
        return -1;
    pRet = ProcAdd (FileName);

    return (pRet);
};

#ifdef __cplusplus
}
#endif
```

```
LIBRARY StegIntf
EXPORTS
    InitCall
    IsTypeCall
    DetectCall
```

```
#ifndef WaveDetector_h
#define WaveDetector_h

#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
using namespace std;

//-------------------------------------------------------------------------------------------
//this class is a child of FileTypeDetector that is used to determine if a file is a
//Wave or not.
//-------------------------------------------------------------------------------------------
class WaveDetector : public FileTypeDetector
{
    private:
        int first_header_length;
        int second_header_length;
        int second_header_offset;
        unsigned char* first_header_magic_bytes;
        unsigned char* second_header_magic_bytes;
        FileDetectionUtilities* utilities;

    public:
        //-------------------------------------------------------------------------------
        //returns true if the file indicated by file_name is a Wave and false otherwise
        //-------------------------------------------------------------------------------
        virtual bool isOfType( const char* file_name );

        //-------------------------------------------------------------------------------
        //contructor for WaveDetector, takes a pointer to the FileDetectionUtilities
        //instance it is to utilize
        //-------------------------------------------------------------------------------
        WaveDetector( FileDetectionUtilities* pointer_to_utilites );
};

#endif
```

```
#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
#include "WaveDetector.h"

using namespace std;

WaveDetector::WaveDetector( FileDetectionUtilities* pointer_to_utilities )
{
    first_header_length = 4;
    second_header_length = 7;
    second_header_offset = 8;
    utilities = pointer_to_utilities;
    first_header_magic_bytes = new unsigned char[first_header_length];
    second_header_magic_bytes = new unsigned char[second_header_length];

    //initialize header_magic_bytes to reflect the magic bytes in a Wave header
    first_header_magic_bytes[0] = 0x52;
    first_header_magic_bytes[1] = 0x49;
    first_header_magic_bytes[2] = 0x46;
    first_header_magic_bytes[3] = 0x46;

    second_header_magic_bytes[0] = 0x57;
    second_header_magic_bytes[1] = 0x41;
    second_header_magic_bytes[2] = 0x56;
    second_header_magic_bytes[3] = 0x45;
    second_header_magic_bytes[4] = 0x66;
    second_header_magic_bytes[5] = 0x6d;
    second_header_magic_bytes[6] = 0x74;
}

bool WaveDetector::isOfType( const char* file_name )
{
    bool is_wave = true;
    if( !utilities->matchesHeader( file_name, first_header_magic_bytes, first_header_length, 0 ) )
        is_wave = false;
    if( !utilities->matchesHeader( file_name, second_header_magic_bytes, second_header_length,
                                   second_header_offset ))
        is_wave = false;
    return is_wave;
}
```

```
#ifndef ZipDetector_h
#define ZipDetector_h

#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
using namespace std;

//-------------------------------------------------------------------------------------------
//this class is a child of FileTypeDetector that is used to determine if a file is a
//Zip or not.
//-------------------------------------------------------------------------------------------
class ZipDetector : public FileTypeDetector
{
    private:
        int header_length;
        unsigned char* header_magic_bytes;
        FileDetectionUtilities* utilities;

    public:
        //-------------------------------------------------------------------------------
        //returns true if the file indicated by file_name is a Zip and false otherwise
        //-------------------------------------------------------------------------------
        virtual bool isOfType( const char* file_name );

        //-------------------------------------------------------------------------------
        //contructor for ZipDetector, takes a pointer to the FileDetectionUtilities
        //instance it is to utilize
        //-------------------------------------------------------------------------------
        ZipDetector( FileDetectionUtilities* pointer_to_utilites );
};

#endif
```

```
#include "FileTypeDetector.h"
#include "FileDetectionUtilities.h"
#include "ZipDetector.h"

using namespace std;

ZipDetector::ZipDetector( FileDetectionUtilities* pointer_to_utilities )
{
    header_length = 4;
    utilities = pointer_to_utilities;
    header_magic_bytes = new unsigned char[header_length];


    //initialize header_magic_bytes to reflect the magic bytes in a Zip header
    header_magic_bytes[0] = 0x50;
    header_magic_bytes[1] = 0x4b;
    header_magic_bytes[2] = 0x03;
    header_magic_bytes[3] = 0x04;
}

bool ZipDetector::isOfType( const char* file_name )
{
    bool is_zip = true;
    if( !utilities->matchesHeader( file_name, header_magic_bytes, header_length, 0 ) )
        is_zip = false;
    return is_zip;
}
```

```
/*
 * Directory.java
 *
 * Created on February 10, 2005, 9:31 AM
 */

package examples.StegKit;


import javax.swing.*;
import java.util.*;
import java.awt.event.*;
import com.jniwrapper.*;
import com.jniwrapper.win32.registry.RegistryKey;

/**
 *
 * @author  bill
 */
public class Directory extends javax.swing.JFrame {

    public Directory() {
        initComponents();
        DirectoryList = new javax.swing.DefaultListModel();
        jList1.setModel(DirectoryList);

        LoadList();
    }

    private void initComponents() {//GEN-BEGIN:initComponents
        java.awt.GridBagConstraints gridBagConstraints;

        jScrollPane1 = new javax.swing.JScrollPane();
        jList1 = new javax.swing.JList();
        jButtonAdd = new javax.swing.JButton();
        jButtonDelete = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jButtonOK = new javax.swing.JButton();

        getContentPane().setLayout(new java.awt.GridBagLayout());

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });

        jScrollPane1.setPreferredSize(new java.awt.Dimension(400, 100));
        jScrollPane1.setViewportView(jList1);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.gridwidth = 11;
        gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 1.0;
```

```
        getContentPane().add(jScrollPane1, gridBagConstraints);

        jButtonAdd.setText("Add");
        jButtonAdd.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonAddActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 5;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        getContentPane().add(jButtonAdd, gridBagConstraints);

        jButtonDelete.setText("Delete");
        jButtonDelete.setPreferredSize(new java.awt.Dimension(72, 25));
        jButtonDelete.setMaximumSize(new java.awt.Dimension(72, 25));
        jButtonDelete.setMinimumSize(new java.awt.Dimension(72, 25));
        jButtonDelete.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonDeleteActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 6;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        getContentPane().add(jButtonDelete, gridBagConstraints);

        jLabel1.setPreferredSize(new java.awt.Dimension(120, 30));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.gridwidth = 5;
        getContentPane().add(jLabel1, gridBagConstraints);

        jButtonOK.setText("OK");
        jButtonOK.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonOKActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 10;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        getContentPane().add(jButtonOK, gridBagConstraints);

        pack();
    }//GEN-END:initComponents

    private void jButtonDeleteActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButtonDeleteActionPerformed
        RegistryKey registryKey;
```

```
        registryKey = RegistryKey.CURRENT_USER.createSubKey("Software\\VB and
VBA Program Settings\\StegKit\\Directory", true);
        for (int i = DirectoryList.size() - 1;i > -1;i--) {
            if (jList1.isSelectedIndex(i) == true) {
                registryKey.values().remove(DirectoryList.elementAt(i));
            }
        }
        LoadList();
    }//GEN-LAST:event_jButtonDeleteActionPerformed

    private void jButtonAddActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButtonAddActionPerformed
        JFileChooser chooser = new JFileChooser();
        String KeyName;
        String TempKey;
        boolean Found = false;

        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        int returnVal = chooser.showOpenDialog(this);
        if(returnVal == JFileChooser.APPROVE_OPTION) {
            RegistryKey registryKey;

            KeyName =chooser.getSelectedFile().getPath();
            KeyName = KeyName.toLowerCase();
            for (int i = 0; i <DirectoryList.size();i++) {
                TempKey = (String)DirectoryList.elementAt(i);
                TempKey = TempKey.toLowerCase();
                if (KeyName.indexOf(TempKey) > -1 ||
                    TempKey.indexOf(KeyName) > -1) {
                    Found = true;
                    String message = "Directory conflict";
                    JOptionPane.showMessageDialog(this, message);

                 }
            }
            if (Found == false) {
                registryKey =
RegistryKey.CURRENT_USER.createSubKey("Software\\VB and VBA Program
Settings\\StegKit\\Directory", true);
                registryKey.values().put(KeyName,"");
            }
        }
        LoadList();
    }//GEN-LAST:event_jButtonAddActionPerformed

    private void jButtonOKActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButtonOKActionPerformed
         this.hide();
        this.dispose();
        java.awt.event.WindowEvent evts = null;
        exitForm(evts);
    }//GEN-LAST:event_jButtonOKActionPerformed

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_exitForm
```

```
        //System.exit(0);
    }//GEN-LAST:event_exitForm
    private void LoadList() {
        RegistryKey registryKey;
        List DirList;
        String KeyName;
        Object fred;

        DirectoryList.clear();
        registryKey = RegistryKey.CURRENT_USER.createSubKey("Software\\VB and
VBA Program Settings\\StegKit\\Directory", true);

        DirList = registryKey.values().getEntries();
        for (int i = 0;i < DirList.size();i++) {
            fred = DirList.get(i);
            KeyName = fred.toString();
            KeyName = KeyName.substring(KeyName.indexOf("Name=") +
5,KeyName.length());
            KeyName = KeyName.substring(0,KeyName.indexOf("type=")- 1);
            DirectoryList.addElement(KeyName);
        }
    }

     public static void main(String args[]) {
        DefaultLibraryLoader.getInstance().addPath("C:\\Jniwrapper\\bin");
        new Directory().show();
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButtonAdd;
    private javax.swing.JButton jButtonDelete;
    private javax.swing.JButton jButtonOK;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JList jList1;
    private javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration//GEN-END:variables
    private javax.swing.DefaultListModel DirectoryList;

}
```

```
<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.0" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">
  <SyntheticProperties>
    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>
  </SyntheticProperties>
  <Events>
    <EventHandler event="windowClosing" listener="java.awt.event.WindowListener"
parameters="java.awt.event.WindowEvent" handler="exitForm"/>
  </Events>
  <AuxValues>
    <AuxValue name="designerSize" type="java.awt.Dimension" value="-84,-
19,0,5,115,114,0,18,106,97,118,97,46,97,119,116,46,68,105,109,101,110,115,105,11
1,110,65,-114,-39,-41,-
84,95,68,20,2,0,2,73,0,6,104,101,105,103,104,116,73,0,5,119,105,100,116,104,120,
112,0,0,1,82,0,0,1,-91"/>
  </AuxValues>

  <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"/>
  <SubComponents>
    <Container class="javax.swing.JScrollPane" name="jScrollPane1">
      <Properties>
        <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
          <Dimension value="[400, 100]"/>
        </Property>
      </Properties>
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagCons
traintsDescription">
          <GridBagConstraints gridX="0" gridY="0" gridWidth="11" gridHeight="1"
fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="0" insetsBottom="0"
insetsRight="0" anchor="10" weightX="1.0" weightY="1.0"/>
        </Constraint>
      </Constraints>

      <Layout
class="org.netbeans.modules.form.compat2.layouts.support.JScrollPaneSupportLayou
t"/>
      <SubComponents>
        <Component class="javax.swing.JList" name="jList1">
        </Component>
      </SubComponents>
    </Container>
    <Component class="javax.swing.JButton" name="jButtonAdd">
      <Properties>
        <Property name="text" type="java.lang.String" value="Add"/>
      </Properties>
      <Events>
        <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="jButtonAddActionPerformed"/>
      </Events>
      <Constraints>
```

```
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagCons
traintsDescription">
          <GridBagConstraints gridX="5" gridY="1" gridWidth="1" gridHeight="1"
fill="0" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="0" insetsBottom="0"
insetsRight="0" anchor="13" weightX="0.0" weightY="0.0"/>
        </Constraint>
      </Constraints>
    </Component>
    <Component class="javax.swing.JButton" name="jButtonDelete">
      <Properties>
        <Property name="text" type="java.lang.String" value="Delete"/>
        <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
          <Dimension value="[72, 25]"/>
        </Property>
        <Property name="maximumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
          <Dimension value="[72, 25]"/>
        </Property>
        <Property name="minimumSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
          <Dimension value="[72, 25]"/>
        </Property>
      </Properties>
      <Events>
        <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="jButtonDeleteActionPerformed"/>
      </Events>
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagCons
traintsDescription">
          <GridBagConstraints gridX="6" gridY="1" gridWidth="1" gridHeight="1"
fill="0" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="0" insetsBottom="0"
insetsRight="0" anchor="13" weightX="0.0" weightY="0.0"/>
        </Constraint>
      </Constraints>
    </Component>
    <Component class="javax.swing.JLabel" name="jLabel1">
      <Properties>
        <Property name="preferredSize" type="java.awt.Dimension"
editor="org.netbeans.beaninfo.editors.DimensionEditor">
          <Dimension value="[120, 30]"/>
        </Property>
      </Properties>
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagCons
traintsDescription">
          <GridBagConstraints gridX="0" gridY="1" gridWidth="5" gridHeight="1"
fill="0" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="0" insetsBottom="0"
insetsRight="0" anchor="10" weightX="0.0" weightY="0.0"/>
```

```
        </Constraint>
      </Constraints>
    </Component>
    <Component class="javax.swing.JButton" name="jButtonOK">
      <Properties>
        <Property name="text" type="java.lang.String" value="OK"/>
      </Properties>
      <Events>
        <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="jButtonOKActionPerformed"/>
      </Events>
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagCons
traintsDescription">
          <GridBagConstraints gridX="10" gridY="1" gridWidth="1" gridHeight="1"
fill="0" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="0" insetsBottom="0"
insetsRight="0" anchor="13" weightX="0.0" weightY="0.0"/>
        </Constraint>
      </Constraints>
    </Component>
  </SubComponents>
</Form>
```

```
/*
 * DllPlugIn.java
 *
 * Created on April 29, 2005, 10:24 AM
 */

package examples.StegKit;

import com.jniwrapper.win32.FunctionName;
import com.jniwrapper.*;

/**
 *
 * @author  bill
 */
public class DllPlugIn extends StegPlugIns {
    private int StegPlugType = 1;
    private String DllName;
    /** Creates a new instance of DllPlugIn */
    public DllPlugIn(String Name) {
        DefaultLibraryLoader.getInstance().addPath("C:\\Jniwrapper\\bin");
        DllName = Name;
    }

    public int PlugType() {
        return StegPlugType;
    }
    public String PlugName() {
        return DllName;
    }

    public int init(StringBuffer TypesList){
        Library StegLib;
        StegLib = new Library(DllName);
        UInt16 ReturnType= new UInt16(0x0);
        AnsiString ReturnString= new AnsiString("
");
        String RetString;
        try {
            Function InitCall = StegLib.getFunction("init");
            InitCall.invoke(ReturnType,ReturnString);
            RetString = ReturnString.toString();
            TypesList.insert(0,RetString);

        }
        catch(FunctionExecutionException e) {

        }

        return (int)ReturnType.getValue();
    }

    public String isOfType(String FileName) {
        AnsiString ReturnString= new AnsiString(256);
        AnsiString FileAnsi = new AnsiString(FileName);
        Library DectLib = new Library(DllName);
        try {
```

```
            Function IsTypeCall = DectLib.getFunction("isOfType");
            IsTypeCall.invoke(ReturnString,FileAnsi);
            return ReturnString.toString();
        }
        catch(FunctionExecutionException e) {
            return "";
        }
    }

    public float detect(String FileName) {
        AnsiString FileAnsi = new AnsiString(FileName);
        SingleFloat FloatRet = new SingleFloat();
        double Percent;
        Library StegLib = new Library(DllName);
        try {
            Function DetectCall = StegLib.getFunction("detect");
            DetectCall.invoke(FloatRet,FileAnsi);
            Percent = FloatRet.getValue();
            return (float) Percent;
        }
        catch(FunctionExecutionException e) {
            return -1;
        }
    }

}
```

```
/*
 * JavaPlugIn.java
 *
 * Created on April 29, 2005, 10:30 AM
 */

package examples.StegKit;

import java.lang.reflect.*;

/**
 *
 * @author  bill
 */
public class JavaPlugIn extends StegPlugIns {
    private int StegPlugType = 2;
    String DllName;

    /** Creates a new instance of JavaPlugIn */
    public JavaPlugIn(String Name) {
        DllName = Name;
   }

    public int PlugType() {
        return StegPlugType;
    }

    public String PlugName() {
        return DllName;
    }

    public int init(StringBuffer TypesList){
      Object javaPlug;
      Class javaDefinition;
      int result;
      try {
          javaDefinition = Class.forName(DllName);
          javaPlug = javaDefinition.newInstance();
          try {
              Method initMethod;
              Class[] parameterTypes = new Class[] {StringBuffer.class};
              Object[] arguments = new Object[] {TypesList};
              Object javaObject;
              initMethod = javaPlug.getClass().getMethod("init",
parameterTypes);
              javaObject = initMethod.invoke(javaPlug, arguments);
              result = ((Integer)javaObject).intValue();

              } catch (NoSuchMethodException e) {
                  result = -1;
              } catch (IllegalAccessException e) {
                  result = -1;
              } catch (InvocationTargetException e) {
                  result = -1;
              }
          } catch (ClassNotFoundException e) {
              result = -1;
```

```
            } catch (IllegalAccessException e) {
                result = -1;
            } catch (IllegalArgumentException e) {
                result = -1;
            } catch (InstantiationException e) {
                result = -1;
            }
        return result;
    }

    public String isOfType(String FileName) {
      Object javaPlug;
      Class javaDefinition;
      String result;
      try {
          javaDefinition = Class.forName(DllName);
          javaPlug = javaDefinition.newInstance();
          try {
              Method initMethod;
              Class[] parameterTypes = new Class[] {String.class};
              Object[] arguments = new Object[] {FileName};
              initMethod = javaPlug.getClass().getMethod("isOfType",
parameterTypes);
              result = (String)initMethod.invoke(javaPlug, arguments);

              } catch (NoSuchMethodException e) {
                  result = "";
              } catch (IllegalAccessException e) {
                  result = "";
              } catch (InvocationTargetException e) {
                  result = "";
              }
          } catch (ClassNotFoundException e) {
              result = "";
          } catch (IllegalAccessException e) {
              result = "";
          } catch (IllegalArgumentException e) {
              result = "";
          } catch (InstantiationException e) {
              result = "";
          }
        return result;
    }


    public float detect(String FileName) {
      Object javaPlug;
      Class javaDefinition;
      float result;
      try {
          javaDefinition = Class.forName(DllName);
          javaPlug = javaDefinition.newInstance();
          try {
              Method initMethod;
              Class[] parameterTypes = new Class[] {String.class};
              Object[] arguments = new Object[] {FileName};
              Object javaObject;
```

```
                initMethod = javaPlug.getClass().getMethod("detect",
parameterTypes);
                javaObject = initMethod.invoke(javaPlug, arguments);
                java.lang.Number numret = (java.lang.Number)javaObject;
                result = numret.floatValue();

                } catch (NoSuchMethodException e) {
                    result = -1;
                } catch (IllegalAccessException e) {
                    result = -1;
                } catch (InvocationTargetException e) {
                    result = -1;
                }
            } catch (ClassNotFoundException e) {
                result = -1;
            } catch (IllegalAccessException e) {
                result = -1;
            } catch (IllegalArgumentException e) {
                result = -1;
            } catch (InstantiationException e) {
                result = -1;
            }
        return result;
    }
}
```

```
/*
 * JGifAna.java
 *
 * Created on April 18, 2005, 2:52 PM
 */

//package examples.JGifAna;
package examples.StegKit;

/**
 *
 * @author  bill
 */
public class JGifAna {

    /** Creates a new instance of JGifAna */
    public JGifAna() {
    }
    public int init(StringBuffer valids) {
        //valids = new String ("gif");
        //valids = valids.substring(0,0);
        valids = valids.append("gif");
        return 1;
    }
     public double detect(String FileName) {

        return 0.72;
    }

     public String isOfType(String FileName){
         String fred = new String("jpg");
         return fred;
     }


}
```

```
/*
 * PluginList.java
 *
 * Created on April 1, 2005, 11:06 AM
 */

package examples.StegKit;

import java.util.Vector;
import com.jniwrapper.*;

/**
 *
 * @author  bill
 */
public class PluginList extends javax.swing.JFrame {

    /** Creates new form PluginList */
    public PluginList(Vector DectTypeList,Vector DectStegList) {
        javax.swing.DefaultListModel listFiles;
        javax.swing.DefaultListModel listFiles2;

        initComponents();

        listFiles = new javax.swing.DefaultListModel();
        jDectList.setModel(listFiles);

        for (int i=0; i<DectTypeList.size(); i++) {
            StegPlugIns PlugIn = (StegPlugIns) DectTypeList.elementAt(i);
            String DllString = PlugIn.PlugName();
            while(DllString.indexOf("\\") > 0 ) {
                DllString = DllString.substring(DllString.indexOf("\\") + 1);
            }

            listFiles.addElement(DllString);
        }
        listFiles2 = new javax.swing.DefaultListModel();
        jStegList.setModel(listFiles2);
        for(int i=0; i< DectStegList.size(); i++) {
            Vector StegTypeDll;
            StegTypeDll = (Vector) DectStegList.elementAt(i);
            String DectType = (String) StegTypeDll.elementAt(0);
            DectType = DectType.concat(": ");
            for (int j=1;j < StegTypeDll.size(); j++) {
                StegPlugIns PlugIn = (StegPlugIns) StegTypeDll.elementAt(j);
                String DllString = PlugIn.PlugName();
                while(DllString.indexOf("\\") > 0 ) {
                    DllString = DllString.substring(DllString.indexOf("\\") +
1);
                }
                DectType = DectType.concat(DllString);
                DectType = DectType.concat ("  ");
            }
            listFiles2.addElement(DectType);
        }
    }
```

```
/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
private void initComponents() {//GEN-BEGIN:initComponents
    java.awt.GridBagConstraints gridBagConstraints;

    jLabel1 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    jDectList = new javax.swing.JList();
    jLabel2 = new javax.swing.JLabel();
    jScrollPane2 = new javax.swing.JScrollPane();
    jStegList = new javax.swing.JList();
    jButtonOk = new javax.swing.JButton();

    getContentPane().setLayout(new java.awt.GridBagLayout());

    setTitle("Pulgins");
    addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent evt) {
            exitForm(evt);
        }
    });

    jLabel1.setText("Detectors");
    jLabel1.setVerticalTextPosition(javax.swing.SwingConstants.TOP);
    getContentPane().add(jLabel1, new java.awt.GridBagConstraints());

    jScrollPane1.setViewportView(jDectList);

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 1;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.weightx = 1.0;
    gridBagConstraints.weighty = 1.0;
    getContentPane().add(jScrollPane1, gridBagConstraints);

    jLabel2.setText("Steganalysis");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 2;
    getContentPane().add(jLabel2, gridBagConstraints);

    jScrollPane2.setViewportView(jStegList);

    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 3;
    gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
    gridBagConstraints.weightx = 1.0;
    gridBagConstraints.weighty = 1.0;
    getContentPane().add(jScrollPane2, gridBagConstraints);

    jButtonOk.setText("OK");
    jButtonOk.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonOkActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        getContentPane().add(jButtonOk, gridBagConstraints);

        pack();
    }//GEN-END:initComponents

    private void jButtonOkActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButtonOkActionPerformed
        this.hide();
        this.dispose();
        // TODO add your handling code here:
    }//GEN-LAST:event_jButtonOkActionPerformed

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_exitForm
        //System.exit(0);
        this.hide();
        this.dispose();

    }//GEN-LAST:event_exitForm

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {

        Vector DectTypeList;
        Vector DectStegList;

        DectTypeList = new Vector();
        DectStegList = new Vector();

        DefaultLibraryLoader.getInstance().addPath("C:\\Jniwrapper\\bin");
        new PluginList(DectTypeList,DectStegList).show();
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButtonOk;
    private javax.swing.JList jDectList;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JList jStegList;
    // End of variables declaration//GEN-END:variables

}
```

```
<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.0" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">
  <Properties>
    <Property name="title" type="java.lang.String" value="Pulgins"/>
  </Properties>
  <SyntheticProperties>
    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>
  </SyntheticProperties>
  <Events>
    <EventHandler event="windowClosing" listener="java.awt.event.WindowListener"
parameters="java.awt.event.WindowEvent" handler="exitForm"/>
  </Events>
  <AuxValues>
    <AuxValue name="designerSize" type="java.awt.Dimension" value="-84,-
19,0,5,115,114,0,18,106,97,118,97,46,97,119,116,46,68,105,109,101,110,115,105,11
1,110,65,-114,-39,-41,-
84,95,68,20,2,0,2,73,0,6,104,101,105,103,104,116,73,0,5,119,105,100,116,104,120,
112,0,0,1,103,0,0,1,127"/>
  </AuxValues>

  <Layout
class="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"/>
  <SubComponents>
    <Component class="javax.swing.JLabel" name="jLabel1">
      <Properties>
        <Property name="text" type="java.lang.String" value="Detectors"/>
        <Property name="verticalTextPosition" type="int" value="1"/>
      </Properties>
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagCons
traintsDescription">
          <GridBagConstraints gridX="-1" gridY="-1" gridWidth="1" gridHeight="1"
fill="0" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="0" insetsBottom="0"
insetsRight="0" anchor="10" weightX="0.0" weightY="0.0"/>
        </Constraint>
      </Constraints>
    </Component>
    <Container class="javax.swing.JScrollPane" name="jScrollPane1">
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagCons
traintsDescription">
          <GridBagConstraints gridX="0" gridY="1" gridWidth="1" gridHeight="1"
fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="0" insetsBottom="0"
insetsRight="0" anchor="10" weightX="1.0" weightY="1.0"/>
        </Constraint>
      </Constraints>

      <Layout
class="org.netbeans.modules.form.compat2.layouts.support.JScrollPaneSupportLayou
t"/>
      <SubComponents>
        <Component class="javax.swing.JList" name="jDectList">
```

```
        </Component>
      </SubComponents>
    </Container>
    <Component class="javax.swing.JLabel" name="jLabel2">
      <Properties>
        <Property name="text" type="java.lang.String" value="Steganalysis"/>
      </Properties>
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagCons
traintsDescription">
          <GridBagConstraints gridX="0" gridY="2" gridWidth="1" gridHeight="1"
fill="0" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="0" insetsBottom="0"
insetsRight="0" anchor="10" weightX="0.0" weightY="0.0"/>
        </Constraint>
      </Constraints>
    </Component>
    <Container class="javax.swing.JScrollPane" name="jScrollPane2">
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagCons
traintsDescription">
          <GridBagConstraints gridX="0" gridY="3" gridWidth="1" gridHeight="1"
fill="1" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="0" insetsBottom="0"
insetsRight="0" anchor="10" weightX="1.0" weightY="1.0"/>
        </Constraint>
      </Constraints>

      <Layout
class="org.netbeans.modules.form.compat2.layouts.support.JScrollPaneSupportLayou
t"/>
      <SubComponents>
        <Component class="javax.swing.JList" name="jStegList">
        </Component>
      </SubComponents>
    </Container>
    <Component class="javax.swing.JButton" name="jButtonOk">
      <Properties>
        <Property name="text" type="java.lang.String" value="OK"/>
      </Properties>
      <Events>
        <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="jButtonOkActionPerformed"/>
      </Events>
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignGridBagLayout$GridBagCons
traintsDescription">
          <GridBagConstraints gridX="0" gridY="4" gridWidth="1" gridHeight="1"
fill="0" ipadX="0" ipadY="0" insetsTop="0" insetsLeft="0" insetsBottom="0"
insetsRight="0" anchor="13" weightX="0.0" weightY="0.0"/>
        </Constraint>
      </Constraints>
```

```
    </Component>
  </SubComponents>
</Form>
```

```
/*
 * StegKit.java
 *
 * Created on January 7, 2005, 2:59 PM
 */


package examples.StegKit;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.lang.Number.*;
import java.util.Vector;
import com.jniwrapper.*;
import com.jniwrapper.win32.Handle;
import com.jniwrapper.win32.system.Kernel32;
import com.jniwrapper.win32.registry.RegistryKey;
import com.jniwrapper.win32.gdi.Icon;

import com.jniwrapper.win32.Msg;
import com.jniwrapper.win32.gdi.Icon;
//import com.jniwrapper.win32.shell.TrayIcon;
import com.jniwrapper.win32.shell.TrayIconListener;
import com.jniwrapper.win32.shell.TrayMessage;
import com.jniwrapper.win32.system.VersionInfo;
import com.jniwrapper.win32.system.DllVersionInfo;
import com.jniwrapper.util.Logger;
import com.jniwrapper.NoSuchFunctionException;


import com.jniwrapper.util.Logger;
import com.jniwrapper.win32.FunctionName;
import com.jniwrapper.win32.LastErrorException;
import com.jniwrapper.win32.Msg;
import com.jniwrapper.win32.Point;
import com.jniwrapper.win32.gdi.Cursor;
import com.jniwrapper.win32.ui.WindowProc;
import com.jniwrapper.win32.ui.Wnd;
import com.jniwrapper.win32.ui.WndClass;
import com.jniwrapper.win32.shell.*;
import javax.swing.Timer;
import javax.swing.JPopupMenu;
import javax.swing.JFrame;
import java.awt.event.*;
import java.awt.Component;
import java.awt.Dimension;
import java.util.*;


/**
 *
 * @author  bill
 */
public class StegKit extends javax.swing.JFrame {
```

```
    public StegKit() {
        File QuarDir;

        initComponents();
        listFiles = new javax.swing.DefaultListModel();
        listDetects = new javax.swing.DefaultListModel();
        DectStegList = new Vector();
        ThreadList = new Vector();
        DectTypeList = new Vector();
        jList1.setModel(listFiles);
        jList2.setModel(listDetects);
        QuarDir = new File(System.getProperty("user.dir") + "\\Quaratine");
        QuarDir.mkdir();
        Load_Dll_list();
        Load_Java_list();
        Get_Quarn_List();

        RegistryKey registryKey;
        String LogFileState;
        registryKey = RegistryKey.CURRENT_USER.createSubKey("Software\\VB and
VBA Program Settings\\StegKit\\Options", true);
        LogFileState = (String)registryKey.values().get("Logfile");
        if (LogFileState == null) {
            mnuLogFile.setState(false);
            registryKey.values().put("Logfile","False");

        }
        else if ( LogFileState.compareTo("False")== 0) {
            mnuLogFile.setState(false);
        }
        Integer Numtemp;
        LogFileState = (String)registryKey.values().get("Limit");
        if (LogFileState == null) {
            Numtemp = new Integer(70);
            registryKey.values().put("Limit","70");
        }
        else {
            Numtemp = new Integer(LogFileState);
        }
        jSlider1.setValue(Numtemp.intValue());


        initTimer();
        timer.start();
        TrayStuff = new StegKit.TrayIconSample();
        if (listDetects.size() > 0)
        {
            TrayStuff.changeIconToMany();
        }
        StartMon();

    }


    private void initComponents() {//GEN-BEGIN:initComponents
        JPopupDelete = new javax.swing.JPopupMenu();
```

```
JMenuDelete = new javax.swing.JMenuItem();
jPopupShow = new javax.swing.JPopupMenu();
jMenuShow = new javax.swing.JMenuItem();
jMenuExit2 = new javax.swing.JMenuItem();
jScrollPane1 = new javax.swing.JScrollPane();
jList1 = new javax.swing.JList();
jScrollPane2 = new javax.swing.JScrollPane();
jList2 = new javax.swing.JList();
jSlider1 = new javax.swing.JSlider();
jMenuBar1 = new javax.swing.JMenuBar();
mnuFile = new javax.swing.JMenu();
mnuHide = new javax.swing.JMenuItem();
jSeparator1 = new javax.swing.JSeparator();
mnuExit = new javax.swing.JMenuItem();
mnuOptions = new javax.swing.JMenu();
mnuDirectories = new javax.swing.JMenuItem();
mnuLogFile = new javax.swing.JCheckBoxMenuItem();
mnuRefresh = new javax.swing.JMenuItem();
mnuStart = new javax.swing.JMenuItem();
mnuStop = new javax.swing.JMenuItem();
mnuDllList = new javax.swing.JMenuItem();
mnuScan = new javax.swing.JMenu();
mnuScanFile = new javax.swing.JMenuItem();
mnuScanDir = new javax.swing.JMenuItem();

JMenuDelete.setText("Delete");
JMenuDelete.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        JMenuDeleteActionPerformed(evt);
    }
});

JPopupDelete.add(JMenuDelete);

jMenuShow.setText("Show StegKit");
jMenuShow.setToolTipText("Show StegKit");
jMenuShow.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuShowActionPerformed(evt);
    }
});

jPopupShow.add(jMenuShow);

jMenuExit2.setText("Exit from StegKit");
jMenuExit2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuExit2ActionPerformed(evt);
    }
});

jPopupShow.add(jMenuExit2);

setTitle("Stegkit");
addMouseMotionListener(new java.awt.event.MouseMotionAdapter() {
    public void mouseMoved(java.awt.event.MouseEvent evt) {
        formMouseMoved(evt);
```

```
    }
});
addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
    }
});

jScrollPane1.setViewportView(jList1);

getContentPane().add(jScrollPane1, java.awt.BorderLayout.NORTH);

jList2.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jList2MouseClicked(evt);
    }
});

jScrollPane2.setViewportView(jList2);

getContentPane().add(jScrollPane2, java.awt.BorderLayout.SOUTH);

jSlider1.setPaintLabels(true);
jSlider1.setPaintTicks(true);
jSlider1.setMinorTickSpacing(1);
jSlider1.setMajorTickSpacing(10);
jSlider1.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseReleased(java.awt.event.MouseEvent evt) {
        jSlider1MouseReleased(evt);
    }
});

getContentPane().add(jSlider1, java.awt.BorderLayout.CENTER);

mnuFile.setText("File");
mnuFile.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuFileActionPerformed(evt);
    }
});

mnuHide.setText("Hide");
mnuHide.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuHideActionPerformed(evt);
    }
});

mnuFile.add(mnuHide);

mnuFile.add(jSeparator1);

mnuExit.setText("Exit");
mnuExit.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuExitActionPerformed(evt);
    }
```

```
});

mnuFile.add(mnuExit);

jMenuBar1.add(mnuFile);

mnuOptions.setText("Options");
mnuDirectories.setText("Directories...");
mnuDirectories.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuDirectoriesActionPerformed(evt);
    }
});

mnuOptions.add(mnuDirectories);

mnuLogFile.setSelected(true);
mnuLogFile.setText("LogFile");
mnuLogFile.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuLogFileActionPerformed(evt);
    }
});

mnuOptions.add(mnuLogFile);

mnuRefresh.setText("Refresh");
mnuRefresh.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuRefreshActionPerformed(evt);
    }
});

mnuOptions.add(mnuRefresh);

mnuStart.setText("Start Monitor");
mnuStart.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuStartActionPerformed(evt);
    }
});

mnuOptions.add(mnuStart);

mnuStop.setText("Stop Monitor");
mnuStop.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuStopActionPerformed(evt);
    }
});

mnuOptions.add(mnuStop);

mnuDllList.setText("Register Dlls");
mnuDllList.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuDllListActionPerformed(evt);
```

```
            }
        });

        mnuOptions.add(mnuDllList);

        jMenuBar1.add(mnuOptions);

        mnuScan.setText("Scan");
        mnuScanFile.setText("Scan a File");
        mnuScanFile.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnuScanFileActionPerformed(evt);
            }
        });

        mnuScan.add(mnuScanFile);

        mnuScanDir.setText("Scan a Directory");
        mnuScanDir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnuScanDirActionPerformed(evt);
            }
        });

        mnuScan.add(mnuScanDir);

        jMenuBar1.add(mnuScan);

        setJMenuBar(jMenuBar1);

        pack();
    }//GEN-END:initComponents

    private void mnuDllListActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_mnuDllListActionPerformed
        new PluginList(DectTypeList,DectStegList).show();
        // TODO add your handling code here:
    }//GEN-LAST:event_mnuDllListActionPerformed

    private void jMenuExit2ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jMenuExit2ActionPerformed
         mnuExitActionPerformed (evt);
    }//GEN-LAST:event_jMenuExit2ActionPerformed

    private void jMenuShowActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jMenuShowActionPerformed
        this.setVisible(true);
    }//GEN-LAST:event_jMenuShowActionPerformed

    private void formMouseMoved(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_formMouseMoved
        // TODO add your handling code here:
    }//GEN-LAST:event_formMouseMoved

    private void mnuLogFileActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_mnuLogFileActionPerformed
        RegistryKey registryKey;
```

```
        registryKey = RegistryKey.CURRENT_USER.createSubKey("Software\\VB and
VBA Program Settings\\StegKit\\Options", true);
        if (mnuLogFile.getState()== true) {
            registryKey.values().put("Logfile","True");
        }
        else {
            registryKey.values().put("Logfile","False");
        }

    }//GEN-LAST:event_mnuLogFileActionPerformed

    private void mnuDirectoriesActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_mnuDirectoriesActionPerformed
        new Directory().show();

    }//GEN-LAST:event_mnuDirectoriesActionPerformed

    private void jList2MouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jList2MouseClicked
        if (evt.getButton() == 3) {
            if (jList2.isSelectionEmpty() == false) {
                JPopupDelete.show(evt.getComponent(), evt.getX(), evt.getY());
            }
        }
    }//GEN-LAST:event_jList2MouseClicked

    private void JMenuDeleteActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_JMenuDeleteActionPerformed
        boolean[] Selected;
        String FileName;
        String FileName2;
        FileWriter fileOut = null;
        PrintWriter printOut = null;
        boolean IsLogFile;

        IsLogFile = mnuLogFile.getState();
        if (IsLogFile) {
            try {
               fileOut = new FileWriter(System.getProperty("user.dir") +
"\\logfile.txt",true);
               printOut = new PrintWriter(fileOut);
            }
            catch (IOException e) {
                IsLogFile = false;
            }
        }

        Selected = new boolean[listDetects.size()];
        for (int i = 0;i < listDetects.size();i++) {
            Selected[i] = jList2.isSelectedIndex(i);
        }

        for (int i = 0;i < listDetects.size();i++) {
            if (jList2.isSelectedIndex(i) == true) {
                FileName = (String) listDetects.elementAt(i);
                FileName = FileName.substring(0,FileName.indexOf(" "));
```

```
                for (int j = 0;j < listDetects.size();j++) {
                    if (Selected[j] == false) {
                        FileName2 = (String) listDetects.elementAt(j);
                        FileName2 = FileName2.substring(0,FileName2.indexOf("
"));
                        if (FileName2.compareTo(FileName)== 0) {
                            Selected[j] = true;
                        }
                    }
                }
                FileName = System.getProperty("user.dir") + "\\Quaratine\\" +
FileName;
                File fileDel = new File(FileName);
                if (fileDel.exists() == true) {
                    fileDel.delete();
                }
                if (IsLogFile) {
                    printOut.println("Deleted: " + FileName);
                }
            }
        }
        for (int i = listDetects.size() - 1;i > -1;i--) {
            if (Selected[i] == true) {
                listDetects.remove(i);
            }
        }
        if (IsLogFile) {
            try {
                printOut.close();
                fileOut.close();
            }
            catch (IOException e) {
            }
        }
        if (listDetects.size() == 0)
        {
            TrayStuff.changeIconToNone();
        }

        Build_Quarn_List();

    }//GEN-LAST:event_JMenuDeleteActionPerformed

    private void mnuScanDirActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_mnuScanDirActionPerformed
        JFileChooser chooser = new JFileChooser();
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        int returnVal = chooser.showOpenDialog(this);
        if(returnVal == JFileChooser.APPROVE_OPTION) {
            File dir = new File(chooser.getSelectedFile().getPath());
            String[] children = dir.list();
            for (int i = 0;i < children.length;i++) {
                children[i] = chooser.getSelectedFile().getPath() + "\\" +
children[i];
                File Isdir = new File(children[i]);
                if (Isdir.isFile() == true) {
                    listFiles.addElement(children[i].toLowerCase());
```

```
                }
            }
        }
    }//GEN-LAST:event_mnuScanDirActionPerformed

    private void mnuScanFileActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_mnuScanFileActionPerformed
        JFileChooser chooser = new JFileChooser();
        int returnVal = chooser.showOpenDialog(this);
        if(returnVal == JFileChooser.APPROVE_OPTION) {
            listFiles.addElement(chooser.getSelectedFile().getPath());
        }
    }//GEN-LAST:event_mnuScanFileActionPerformed

    private void mnuRefreshActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_mnuRefreshActionPerformed
        Refresh_List();
    }//GEN-LAST:event_mnuRefreshActionPerformed

    private void Refresh_List() {
        boolean Found;
        int fred;
        boolean[] Selected;
        String FileName;
        File dir = new File(System.getProperty("user.dir") + "\\Quaratine");
        String[] children = dir.list();
        Selected = new boolean[listDetects.size()];
        for (int i = 0;i < listDetects.size();i++) {
            Selected[i] = false;
        }
        for (int i = 0;i < children.length;i++) {
            children[i] =children[i].toLowerCase();
            if (children[i].compareTo("quarn.txt") != 0) {
                Found = false;
                for (int j= 0;j < listDetects.size();j++) {
                    FileName = (String) listDetects.elementAt(j);
                    FileName = FileName.substring(0,FileName.indexOf(" "));
                    if (children[i].compareTo(FileName)== 0) {
                        Selected[j] = true;
                        Found = true;
                    }
                }
                if (Found == false) {
                    String FileDest = System.getProperty("user.dir") +
"\\Quaratine\\" + children[i];
                    File fileDel = new File(FileDest);
                    fileDel.delete();
                }
            }
        }
        for (int i = listDetects.size() - 1;i > -1;i--) {
            if (Selected[i] == false) {
                listDetects.remove(i);
            }
        }
        if (listDetects.size() == 0)
        {
```

```
            TrayStuff.changeIconToNone();
        }
        Build_Quarn_List();

    }


    private void Build_Quarn_List() {
        try {
            FileWriter fileQuarn = new FileWriter(System.getProperty("user.dir")
+ "\\Quaratine\\quarn.txt");
            PrintWriter printQuarn = new PrintWriter(fileQuarn);
            for (int i = 0;i < listDetects.size();i++) {
                printQuarn.println((String) listDetects.elementAt(i));
            }
            printQuarn.close();
            fileQuarn.close();
        }
        catch (IOException e) {
        }

    }
    private void jSlider1MouseReleased(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jSlider1MouseReleased
        String Temp;
        int num;
        boolean found = false;
        RegistryKey registryKey;
        int DeleteState = 2;
        Integer OrgState;


        registryKey = RegistryKey.CURRENT_USER.createSubKey("Software\\VB and
VBA Program Settings\\StegKit\\Options", true);
        String LimitState = (String)registryKey.values().get("Limit");
        OrgState = new Integer(LimitState);

        for (int i = listDetects.size() - 1;i > -1;i--) {
            Temp = (String) listDetects.elementAt(i);
            while(Temp.indexOf(" ") > -1) {
                Temp = Temp.substring(Temp.indexOf(" ") + 1);
            }
            Temp = Temp.substring(0,Temp.length()- 1);
            Integer Numtemp = new Integer(Temp);
            num = Numtemp.intValue();
            if (num < jSlider1.getValue()) {
                if (DeleteState == 2) {
                    DeleteState = JOptionPane.showConfirmDialog(null,"Delete
Files?", "Delete Files?", JOptionPane.YES_NO_OPTION);
                }
                if (DeleteState == 0) {
                    listDetects.remove(i);
                    found = true;
                }
            }
        }
        if (found == true) {
```

```
            Refresh_List();
        }
        if (DeleteState == 1) {
            jSlider1.setValue(OrgState.intValue());
        }
        else {
           Temp = String.valueOf(jSlider1.getValue());
           registryKey.values().put("Limit",Temp);
        }
    }//GEN-LAST:event_jSlider1MouseReleased

    private void mnuStopActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_mnuStopActionPerformed
        //Watcher = false;
        Watcher = false;
        Thread dirThread = null;
        for(int i=0; i< ThreadList.size(); i++) {
            dirThread = (Thread) ThreadList.elementAt(i);
            dirThread.interrupted();
        }
        mnuStart.setEnabled(true);
        mnuStop.setEnabled(false);

    }//GEN-LAST:event_mnuStopActionPerformed

    private void mnuStartActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_mnuStartActionPerformed
        StartMon();
    }//GEN-LAST:event_mnuStartActionPerformed

     private void StartMon() {
        RegistryKey registryKey;
        Thread dirThread = null;
        java.util.List DirList;
        Object fred;
        String KeyName;

        Watcher = true;
        ThreadList.clear();
        registryKey = RegistryKey.CURRENT_USER.createSubKey("Software\\VB and
VBA Program Settings\\StegKit\\Directory", true);

        DirList = registryKey.values().getEntries();
        for (int i = 0;i < DirList.size();i++) {
            fred = DirList.get(i);
            KeyName = fred.toString();
            KeyName = KeyName.substring(KeyName.indexOf("Name=") +
5,KeyName.length());
            KeyName = KeyName.substring(0,KeyName.indexOf("type=")- 1);
            dirThread = new GetNewFile(KeyName);
            dirThread.start();
            ThreadList.add(dirThread);
        }
        mnuStart.setEnabled(false);
        mnuStop.setEnabled(true);
     }
```

```
    private void initTimer() {
        timer = new Timer(1000, new ActionListener(){
            public void actionPerformed (ActionEvent evt) {
                FileWriter fileOut = null;
                PrintWriter printOut = null;
                boolean IsLogFile;
                boolean StegFound;
                boolean ErrorFound;
                String DectionTypes;
                String FileName;
                String FileDest;
                String FileRoot;

                IsLogFile = mnuLogFile.getState();

                if (IsLogFile) {
                    try {
                       fileOut = new FileWriter(System.getProperty("user.dir")
+ "\\logfile.txt",true);
                       printOut = new PrintWriter(fileOut);
                    }
                    catch (IOException e) {
                        IsLogFile = false;
                    }

                }
                while (listFiles.size() > 0) {
                    StegFound = false;
                    ErrorFound = false;
                    DectionTypes = "";
                    FileName = (String)listFiles.get(0);
                    FileDest = (String)listFiles.get(0);
                    while(FileDest.indexOf("\\") > 0 ) {
                        FileDest = FileDest.substring(FileDest.indexOf("\\") +
1);
                    }
                    FileRoot = FileDest;
                    FileDest = System.getProperty("user.dir") + "\\Quaratine\\"
+ FileDest;
                    try {
                       CopyTheFile(FileName,FileDest);
                    }
                    catch (IOException e) {
                    }
                    for (int i=0; i<DectTypeList.size(); i++) {
                       StegPlugIns PlugIn;
                       String DectRet;
                       PlugIn = (StegPlugIns) DectTypeList.elementAt(i);
                       DectRet = PlugIn.isOfType(FileName);
                       if (DectRet.length() > 0) {
                           if (DectionTypes.length() == 0) {
                               DectionTypes = DectRet;
                           }
                           else {
                              if (DectionTypes.indexOf(DectRet) == -1 ) {
                                 DectionTypes = DectionTypes.concat(":" +
DectRet);
```

```
                        }
                    }
                }
            }
            while(DectionTypes.length() > 0) {
                String StegType;
                if (DectionTypes.indexOf(":") > -1 ) {
                    StegType =
DectionTypes.substring(0,DectionTypes.indexOf(":"));
                    DectionTypes =
DectionTypes.substring(DectionTypes.indexOf(":") + 1);
                }
                else
                {
                    StegType = DectionTypes;
                    DectionTypes = "";
                }
                StegType = StegType.toLowerCase();

                Vector StegTypeDll;
                for(int i=0; i< DectStegList.size(); i++) {
                   StegTypeDll = (Vector) DectStegList.elementAt(i);
                   if (StegTypeDll.elementAt(0).equals(StegType) ==
true) {
                      for (int j=1;j < StegTypeDll.size(); j++) {
                         float Percent;
                         int intPercent;

                         StegPlugIns PlugIn = (StegPlugIns)
StegTypeDll.elementAt(j);
                         Percent = PlugIn.detect(FileName);
                         if (Percent == -1) {
                             StegTypeDll.remove(j);
                             j = j - 1;
                         }
                         else
                         {
                            Percent = 100 * Percent;
                            intPercent = (int)Percent;
                            if (Percent >= jSlider1.getValue()) {
                               listDetects.addElement(FileRoot + " " +
intPercent + "%");
                               TrayStuff.changeIconToMany();
                               StegFound = true;
                               if (IsLogFile) {
                                  printOut.println("Dectected from " +
PlugIn.PlugName() + ": " + FileName + " " + intPercent + "%");
                               }
                               try {
                                  FileWriter fileQuarn = new
FileWriter(System.getProperty("user.dir") + "\\Quaratine\\quarn.txt",true);
                                  PrintWriter printQuarn = new
PrintWriter(fileQuarn);
                                  printQuarn.println(FileName + " " +
intPercent + "%");
                                  printQuarn.close();
                                  fileQuarn.close();
```

```
                                    }
                                    catch (IOException e) {
                                    }
                                }
                            }
                        }
                    }
                    if (StegFound == false) {
                        if (IsLogFile) {
                            printOut.println("Processed: " + FileName);
                        }
                       File fileDel = new File(FileDest);
                       fileDel.delete();
                    }
                    listFiles.remove(0);
                }
                if (IsLogFile) {
                    try {
                        printOut.close();
                        fileOut.close();
                    }
                    catch (IOException e) {
                    }
                }
            }
        });
    }

    private void Load_Dll_list(){
        File dir = new File(System.getProperty("user.dir"));

        FilenameFilter filter = new FilenameFilter() {
            public boolean accept(File dir, String name) {
                String lowname = name.toLowerCase();
                return lowname.endsWith(".dll");
            }
        };
        String[] children = dir.list(filter);
        for (int i=0; i<children.length; i++) {
            if (children[i].compareTo("jniwrap.dll") != 0)
            {
                String filename = System.getProperty("user.dir") + "\\" +
children[i];
                DllPlugIn  DllPlug = new DllPlugIn(filename);
                StringBuffer RetString = new StringBuffer(255);
                int DllType = DllPlug.init(RetString);

                if (DllType == 0)
                {
                    DectTypeList.add(DllPlug);
                }
                else
                {
                    while (RetString.length()> 0)
                    {
```

```
                        String StegType;
                        if (RetString.indexOf(":") > -1 )
                        {
                            StegType =
RetString.substring(0,RetString.indexOf(":"));
                            RetString =
RetString.delete(0,RetString.indexOf(":")+ 1);
                        }
                        else
                        {
                            StegType = RetString.toString();
                            RetString = RetString.delete(0,RetString.length() +
1);
                        }
                        StegType = StegType.toLowerCase();
                        if (CheckForType(StegType,DllPlug)== false)
                        {
                            Vector StegTypeDll = new Vector();
                            StegTypeDll.addElement(StegType);
                            StegTypeDll.addElement(DllPlug);
                            DectStegList.add(StegTypeDll);
                        }
                    }
                }
            }
        }
    }

    private void Load_Java_list(){
        String JavaName;
        try {
            FileReader fileConfig = new
FileReader(System.getProperty("user.dir") + "\\StegKit.cfg");
            BufferedReader readConfig = new BufferedReader(fileConfig);
            JavaName = readConfig.readLine();
            while (JavaName != null) {
            {
                JavaPlugIn  JavaPlug = new JavaPlugIn(JavaName);
                StringBuffer RetString = new StringBuffer(255);
                int JavaType = JavaPlug.init(RetString);

                if (JavaType == 0)
                {
                    DectTypeList.add(JavaPlug);
                }
                else
                {
                    while (RetString.length()> 0)
                    {
                        String StegType;
                        if (RetString.indexOf(":") > -1 )
                        {
                            StegType =
RetString.substring(0,RetString.indexOf(":"));
                            RetString =
RetString.delete(0,RetString.indexOf(":"));
                        }
```

```
                        else
                        {
                            StegType = RetString.toString();
                            RetString = RetString.delete(0,RetString.length() +
1);
                        }
                        StegType = StegType.toLowerCase();
                        if (CheckForType(StegType,JavaPlug)== false)
                        {
                            Vector StegTypeDll = new Vector();
                            StegTypeDll.addElement(StegType);
                            StegTypeDll.addElement(JavaPlug);
                            DectStegList.add(StegTypeDll);
                        }
                    }
                }

            }
                JavaName = readConfig.readLine();
            }
            readConfig.close();
            fileConfig.close();
        }
        catch (IOException e) {
        }

    }
    private boolean CheckForType(String StegType, StegPlugIns PlugIn) {
        Vector StegTypeDll;
        boolean Found;

        Found = false;
        for(int i=0; i< DectStegList.size(); i++) {
            StegTypeDll = (Vector) DectStegList.elementAt(i);
            if (StegTypeDll.elementAt(0).equals(StegType) == true) {
                Found = true;
                StegTypeDll.addElement(PlugIn);
            }
        }
        return (Found);
    }

    private void CopyTheFile(String srcFile, String dstFile) throws IOException
{

        File src = new File(srcFile);
        File dst = new File(dstFile);

        InputStream in = new FileInputStream(src);
        OutputStream out = new FileOutputStream(dst);

        byte[] buf = new byte[1024];
        int len;
        while ((len = in.read(buf)) > 0) {
            out.write(buf, 0, len);
        }
        in.close();
```

```
        out.close();
    }

    private void mnuExitActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_mnuExitActionPerformed
        TrayStuff.RemoveIcon();
        System.exit(0);
    }//GEN-LAST:event_mnuExitActionPerformed

    private void mnuFileActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_mnuFileActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_mnuFileActionPerformed

    private void mnuHideActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_mnuHideActionPerformed
        this.setVisible(false);
    }//GEN-LAST:event_mnuHideActionPerformed

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {//GEN-
FIRST:event_exitForm
        TrayStuff.RemoveIcon();
        System.exit(0);
    }//GEN-LAST:event_exitForm


    private class GetNewFile extends Thread {
        static final int FILE_LIST_DIRECTORY = 0x1;
        static final int FILE_SHARE_READ = 0x00000001;
        static final int OPEN_EXISTING = 3;
        static final int FILE_FLAG_BACKUP_SEMANTICS = 0x02000000;
        static final int FILE_NOTIFY_CHANGE_FILE_NAME = 0x1;
        static final int FILE_ACTION_ADDED = 0x1;


        class CSecurityAttributes extends Structure
        {
           protected UInt32 Len = new UInt32(0);
           protected UInt32 Des = new UInt32(0);
           protected UInt32 Inher = new UInt32(0);

        public CSecurityAttributes()
        {
            init(new Parameter[]{Len, Des, Inher});
        }
        }



        public GetNewFile(String Directory) {
            super(Directory);
        }

        public void run () {

            Handle _directory = new Handle();
```

```
            CSecurityAttributes SecAttr = new CSecurityAttributes();

            File dir = new File(getName());
            if (!dir.exists()) return;

            final Kernel32 kernel32 = Kernel32.getInstance();
            final Function functionCreateFile =
kernel32.getFunction("CreateFileA");
            functionCreateFile.invoke(_directory, new Parameter[]
            {
                new AnsiString(getName()),
                new UInt32(FILE_LIST_DIRECTORY),
                new UInt32(FILE_SHARE_READ),
                new Pointer(SecAttr),
                new UInt32(OPEN_EXISTING),
                new UInt32(FILE_FLAG_BACKUP_SEMANTICS),
                new Pointer(null, true)
            });
            final long handleValue = _directory.getValue();
            if (_directory.isNull() || handleValue == -1)
                return;
            do {
                final Function functionReadDirectoryChanges =
kernel32.getFunction("ReadDirectoryChangesW");
                Int returnValue = new Int();
                int bufferSize = 1024 * 64;
                PrimitiveArray buffer = new PrimitiveArray(UInt8.class,
bufferSize);
                UInt32 bytesReturned = new UInt32();

                functionReadDirectoryChanges.invoke(returnValue, new Parameter[]
                {
                    _directory,
                    new Pointer(buffer),
                    new UInt32(bufferSize),
                    new Bool(true),
                    new UInt32(FILE_NOTIFY_CHANGE_FILE_NAME),
                    new Pointer(bytesReturned),
                    new Pointer(null, true),
                    new Pointer(null, true)
                });
                long value = returnValue.getValue();
                if (value > 0){
                    int index = 0;
                    int nextEntryIndex = 0;
                    boolean lastEvent = false;
                    int actionInfos;
                    UInt32 result = new UInt32();
                    while (index < bufferSize && !lastEvent)
                    {
                        int nextEntryOffset = readDWORD(buffer.getBytes(),
index);
                        nextEntryIndex += nextEntryOffset;
                        index += 4;
                        lastEvent = nextEntryOffset == 0;
                        int action = readDWORD(buffer.getBytes(), index);
```

```
                        if (action == FILE_ACTION_ADDED) {
                            index += 4;
                            int fileNameLength = readDWORD(buffer.getBytes(),
index);
                            index += 4;
                            String FileName;
                            FileName = getName() + "\\";
                            for (int i = 0; i < fileNameLength >> 1; i++)
                            {
                                char c = readWCHAR(buffer.getBytes(), index);
                                index += 2;
                                FileName = FileName + c;
                            }
                            listFiles.addElement(FileName);
                        }
                        index = nextEntryIndex;
                    }
                }

            } while (Watcher == true);
            final Function function =
Kernel32.getInstance().getFunction("CloseHandle");
            function.invoke(null, _directory);
        }
    }

    private int readDWORD(byte[] buffer, int offset)
    {
        UInt32 result = new UInt32();
        result.read(buffer, offset);
        return (int)result.getValue();
    }

    private char readWCHAR(byte[] buffer, int offset)
    {
        WideChar result = new WideChar();
        result.read(buffer, offset);
        return result.getValue();
    }

    public void Get_Quarn_List()
    {
        String FileName;
        try {
            FileReader fileQuarn = new FileReader(System.getProperty("user.dir")
+ "\\Quaratine\\quarn.txt");
            BufferedReader readQuarn = new BufferedReader(fileQuarn);
            FileName = readQuarn.readLine();
            while (FileName != null) {
                while (FileName.indexOf("\\") > -1 ) {
                    FileName = FileName.substring(FileName.indexOf("\\") + 1);
                }
                listDetects.addElement(FileName);
                FileName = readQuarn.readLine();
            }
            readQuarn.close();
            fileQuarn.close();
```

```
        }
        catch (IOException e) {
        }

    }

//WinPack for JNIWrapper General License
//
//Copyright (c) 2002-2004 MIIK Ltd. All rights reserved.
//
//Redistribution and use in source and binary forms, with or
without
//modification, are permitted provided that the following conditions are met:
//
//  1. Redistributions of source code must retain the copyright notice, this
list
//     of conditions and the following disclaimer.
//
//  2. Redistributions in binary form must reproduce the above copyright
notice,
//     this list of conditions and the following disclaimer in the
documentation
//     and/or other materials provided with the distribution.
//
//THIS SOFTWARE IS PROVIDED BY THE MIIK LTD ``AS IS'' AND ANY EXPRESS OR
IMPLIED
//WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF
//MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN
NO
//EVENT SHALL MIIK LTD OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,
INDIRECT,
//INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT
//LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
OR
//PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
OF
//LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE
//OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF
//ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

public class TrayIconSample
{    private static final String MENU_ITEM_SHOW_TRAY_ICON = "Show Tray Icon";
     private static final String MENU_ITEM_SHOW_STANDARD_ICON = "Show Standard
System Icon";
     private static final String MENU_ITEM_SHOW_CUSTOM_ICON = "Show Custom
Icon";
     private static final String MENU_ITEM_SHOW_CALLOUT = "Show Tray Balloon";
     private static final String MENU_HIDE_TRAY_ICON = "Hide Tray Icon";
     private JLabel lblAdvisoryText;
     private JButton btnShowTrayIcon;
     private JButton btnShowStandardSystemIcon;
     private JButton btnShowCustomIcon;
     private JButton btnShowTrayBallon;
```

```
    private JButton btnHideTrayIcon;
    private TrayIcon _trayIcon;
    private JPopupMenu _popupMenu;
    private JLabel lblNotSupported;
    private boolean _isVersionOK;

    public TrayIconSample()
    {
        initialize();
    }

    public void initialize()
    {
        File dir = new File(System.getProperty("user.dir") + "\\Steg.ico");
        Icon customIcon = new Icon(dir);
        _trayIcon = new TrayIcon(customIcon);
        _trayIcon.setToolTip("StegKit");
        _trayIcon.setPopupMenu(jPopupShow);
    }

    private void changeIconToNone()
    {
        File dir = new File(System.getProperty("user.dir") + "\\Steg.ico");
        Icon customIcon = new Icon(dir);
        _trayIcon.setIcon(customIcon);
    }
   private void changeIconToMany()
    {
        File dir = new File(System.getProperty("user.dir") +
"\\StegFound.ico");
        Icon customIcon = new Icon(dir);
        _trayIcon.setIcon(customIcon);
    }
   private void RemoveIcon()
    {
            _trayIcon.dispose();
    }
    private void showCallout()
    {
        _trayIcon.showMessage(new TrayMessage.Warning("Tray Balloon",
"This program demonstrates the WinPack for JNIWrapper library abilities:\n" +
"\t - tray icon\n" +                  "\t - menu for tray icon\n" +
"\t - balloons from tray icon\n" +                  "etc."));
    }

    public void deactivate()
    {
        _trayIcon.setVisible(false);
    }

    public void activate() throws Exception
    {
        _trayIcon.setVisible(true);
    }
}

class NotifyIconData extends Structure
```

```
{
    private UInt32 _cbSize = new UInt32();
    private Pointer.Void _hWnd = new Pointer.Void();
    private UInt32 _uID = new UInt32();
    private UInt32 _uFlags = new UInt32();
    private UInt32 _uCallbackMessage = new UInt32();
    private Icon _hIcon = new Icon();
    private Str _szTip;
    private UInt32 _dwState = new UInt32();
    private UInt32 _dwStateMask = new UInt32();
    private Str _szInfo;
    private UInt _uTimeout = new UInt();
    private UInt _uVersion = new UInt();
    private Union _union = new Union(new Parameter[]{_uTimeout, _uVersion});
    private Str _szInfoTitle;
    private UInt32 _dwInfoFlags = new UInt32();

    public NotifyIconData(long hwnd, int id)
    {
        initStringParameters();
        VersionInfo versionInfo = new VersionInfo();
        if (versionInfo.isWin2k() | versionInfo.isWinMe())
            init(new Parameter[]{_cbSize, _hWnd, _uID, _uFlags,
_uCallbackMessage, _hIcon, _szTip,
                                 _dwState, _dwStateMask, _szInfo, _union,
_szInfoTitle, _dwInfoFlags});
        else
            init(new Parameter[]{_cbSize, _hWnd, _uID, _uFlags,
_uCallbackMessage, _hIcon, _szTip});
        _cbSize.setValue(getLength());
        _hWnd.setValue(hwnd);
        _uID.setValue(id);
    }

    private void initStringParameters()
    {
        _szTip = new Str("", 128);
        _szInfo = new Str("", 256);
        _szInfoTitle = new Str("", 64);
    }

    public void setCallbackMessage(int callbackMessage)
    {
        _uCallbackMessage.setValue(callbackMessage);
    }

    public void setFlags(long flags)
    {
        _uFlags.setValue(flags);
    }

    public void setIcon(Icon icon)
    {
        _hIcon.setValue(icon.getValue());
    }

    public void setToolTip(String tip)
```

```
        {
            _szTip.setValue(tip);
        }

        public void setState(long value)
        {
            _dwState.setValue(value);
        }

        public void setStateMask(long value)
        {
            _dwStateMask.setValue(value);
        }

        public void setInfo(String value)
        {
            _szInfo.setValue(value);
        }

        public void setTimeout(long value)
        {
            _uTimeout.setValue(value);
        }

        public void setInfoTitle(String value)
        {
            _szInfoTitle.setValue(value);
        }

        public void setInfoFlags(long value)
        {
            _dwInfoFlags.setValue(value);
        }
    }


    public class TrayIcon extends Component
    {

        FunctionName FUNCTION_SHELL_NOTIFY_ICON = new
    FunctionName("Shell_NotifyIcon");
        /*
         * Operation constants for adding, modifying and removing icon in a tray.
         */
        // operation ID for adding tray icon to tray
        static final int NIM_ADD = 0x00000000;
        // operation ID for modifying icon in the tray
        static final int NIM_MODIFY = 0x00000001;
        // operation ID for removig icon from the tray
        static final int NIM_DELETE = 0x00000002;

        /*
         * Constants for configuring a tray icon.
         */
        static final int NIF_MESSAGE = 0x00000001;
        static final int NIF_ICON = 0x00000002;
        static final int NIF_TIP = 0x00000004;
```

```
static final int NIF_STATE = 0x00000008;
static final int NIF_INFO = 0x00000010;

/*
 * Extended tray icon attributes.
 */
static final int NIS_HIDDEN = 0x00000001;
static final int NIS_SHAREDICON = 0x00000002;
public static final int WM_TRAY = Msg.WM_USER + 1;

/**
 * Balloon events
 */
private static final int NIN_BALLOONSHOWN = Msg.WM_USER + 2;
private static final int NIN_BALLOONHIDE = Msg.WM_USER + 3;
private static final int NIN_BALLOONTIMEOUT = Msg.WM_USER + 4;
private static final int NIN_BALLOONUSERCLICK = Msg.WM_USER + 5;

static final String CLASS_NAME = "JW_TrayWindowClassName";

private Map _messageHandlers = new HashMap();
private Object _trayWindowLock = new Object();
private int _curID = 0;
private long _hWnd;

private final int _trayID;
private boolean _disposed = false;
private java.util.List _listeners = new ArrayList();

private boolean _hidingAvailable = false;
private boolean _visible = false;
private Icon _icon;
private JPopupMenu _popupMenu = null;
private JFrame _popupWindow = null;
private TrayIconListener _popupTrayIconListener = null;

public TrayIcon()
{
    this(null);
}

 public TrayIcon(Icon icon)
{
    DllVersionInfo dllVersionInfo = new DllVersionInfo("Comctl32");

    _hidingAvailable = dllVersionInfo.getMajorVersion() >= 5;

    _trayID = _curID++;
    ensureEventProcessing();
    _messageHandlers.put(new Integer(_trayID), this);
    NotifyIconData notifyicondata = new NotifyIconData(_hWnd, _trayID);
    notifyicondata.setCallbackMessage(WM_TRAY);
    notifyicondata.setFlags(NIF_ICON | NIF_MESSAGE | NIF_TIP | NIF_INFO);
    notify(NIM_ADD, notifyicondata);
    _visible = true;
    setIcon(icon);
}
```

```
    private void notify(int operation, NotifyIconData notifyIconData)
    {
        if (_disposed)
        {
            throw new IllegalStateException("Already disposed");
        }
        Function function =
Shell32.getInstance().getFunction(FUNCTION_SHELL_NOTIFY_ICON.toString());
        Bool result = new Bool();
        long errorCode = function.invoke(result, new Int32(operation), new
Pointer(notifyIconData));
        if (!result.getValue())
        {
            throw new LastErrorException(errorCode, "Icon operation failed.");
        }
    }

    private void ensureEventProcessing()
    {
        synchronized (_trayWindowLock)
        {
            if (_hWnd == 0)
            {
                new Thread()
                {
                    public void run()
                    {
                        createEmptyNativeWindow();
                        Wnd.eventLoop(_hWnd);
                    }
                }.start();
                try
                {
                    _trayWindowLock.wait();
                }
                catch (InterruptedException e)
                {
                }
                if (_hWnd == 0)
                {
                    throw new RuntimeException("Event processing window creation
failed");
                }
            }
        }
    }

    public void setIcon(Icon icon)
    {
        if (icon != null && !icon.isNull())
        {
            _icon = icon;
            NotifyIconData notifyicondata = new NotifyIconData(_hWnd, _trayID);
            notifyicondata.setIcon(icon);
            notifyicondata.setFlags(NIF_ICON);
            notify(NIM_MODIFY, notifyicondata);
```

```
    }
}

public void showMessage(TrayMessage value)
{
    NotifyIconData notifyicondata = new NotifyIconData(_hWnd, _trayID);
    notifyicondata.setFlags(NIF_INFO);
    notifyicondata.setInfoTitle(value.getTitle());
    notifyicondata.setInfo(value.getMessage());
    notifyicondata.setTimeout(value.getTimeout() * 1000);
    notifyicondata.setInfoFlags(value.getIconType());
    notify(NIM_MODIFY, notifyicondata);
}

public void dispose()
{
    if (!_disposed)
    {
        notify(NIM_DELETE, new NotifyIconData(_hWnd, _trayID));
        _messageHandlers.remove(new Integer(_trayID));
        _disposed = true;
    }
}

public void setToolTip(String tip)
{
    NotifyIconData notifyicondata = new NotifyIconData(_hWnd, _trayID);
    notifyicondata.setToolTip(tip);
    notifyicondata.setFlags(NIF_TIP);
    notify(NIM_MODIFY, notifyicondata);
}

public void setPopupMenu(JPopupMenu popupMenu)
{
    _popupMenu = popupMenu;

    if (_popupWindow == null)
    {
        _popupWindow = new JFrame();
        _popupWindow.setLocation(-300, -300);
        _popupWindow.pack();
        Wnd wnd = new Wnd(_popupWindow);
        wnd.setTopmost(true);
        wnd.setWindowExStyle(Wnd.WS_EX_TOOLWINDOW);
        _popupWindow.show();
        _popupWindow.addWindowListener(new WindowAdapter()
        {
            public void windowDeactivated(WindowEvent e)
            {
                _popupMenu.setVisible(false);
            }
        });
    }

    if (_popupTrayIconListener == null)
    {
        _popupTrayIconListener = new TrayIconListener()
```

```
            {
                public void trayActionPerformed(long message, int x, int y)
                {
                    if (message == Msg.WM_RBUTTONUP)
                    {
                        if (_popupMenu.getSize().equals(new Dimension(0, 0)))
                        {
                            int px = x - (int)_popupWindow.getLocation().getX();
                            int py =  y -
(int)_popupWindow.getLocation().getY();
                            _popupMenu.show(_popupWindow, px, py);
                            _popupMenu.repaint();
                            _popupMenu.setLocation(x -
(int)_popupMenu.getSize().getWidth(),
                                    y - (int)_popupMenu.getSize().getHeight());
                        }
                        else
                        {
                            int px = x - (int)_popupWindow.getLocation().getX()
- (int)_popupMenu.getSize().getWidth();
                            int py =  y - (int)_popupWindow.getLocation().getY()
- (int)_popupMenu.getSize().getHeight();
                            _popupMenu.show(_popupWindow, px, py);
                            _popupMenu.repaint();
                        }

                        _popupWindow.show();
                    }
                }
            };
        }

        addTrayListener(_popupTrayIconListener);
    }

    public JPopupMenu getPopupMenu()
    {
        return _popupMenu;
    }

     public void removePopupMenu()
    {
        _popupMenu = null;
        removeTrayListener(_popupTrayIconListener);
    }

    /**
     * Adds a specified tray icon listener.
     *
     * @param listener
     */
    public void addTrayListener(TrayIconListener listener)
    {
        _listeners.add(listener);
    }

    public void addTrayListener(MouseListener listener)
```

```
    {
        _listeners.add(listener);
    }

   public void addTrayListener(BalloonListener listener)
    {
        _listeners.add(listener);
    }

     public void removeTrayListener(TrayIconListener listener)
    {
        _listeners.remove(listener);
    }

     public void removeTrayListener(MouseListener listener)
    {
        _listeners.remove(listener);
    }

    public void removeTrayListener(BalloonListener listener)
    {
        _listeners.remove(listener);
    }

    private MouseEvent getMouseEvent(int message, int x, int y)
    {
        MouseEvent mouseEvent;
        switch (message)
        {
            case Msg.WM_LBUTTONDBLCLK:
                mouseEvent = new MouseEvent(this, _trayID,
System.currentTimeMillis(),
                                0, x, y, 2, false, MouseEvent.BUTTON1);
                break;
            case Msg.WM_RBUTTONDBLCLK:
                mouseEvent = new MouseEvent(this, _trayID,
System.currentTimeMillis(),
                                0, x, y, 2, false, MouseEvent.BUTTON2);
                break;
            case Msg.WM_MBUTTONDBLCLK:
                mouseEvent = new MouseEvent(this, _trayID,
System.currentTimeMillis(),
                                0, x, y, 2, false, MouseEvent.BUTTON3);
                break;
            case Msg.WM_LBUTTONDOWN:
            case Msg.WM_LBUTTONUP:
                mouseEvent = new MouseEvent(this, _trayID,
System.currentTimeMillis(),
                                0, x, y, 1, false, MouseEvent.BUTTON1);
                break;
            case Msg.WM_RBUTTONDOWN:
            case Msg.WM_RBUTTONUP:
                mouseEvent = new MouseEvent(this, _trayID,
System.currentTimeMillis(),
                                0, x, y, 1, false, MouseEvent.BUTTON2);
                break;
            case Msg.WM_MBUTTONDOWN:
```

```
            case Msg.WM_MBUTTONUP:
                mouseEvent = new MouseEvent(this, _trayID,
System.currentTimeMillis(),
                                0, x, y, 1, false, MouseEvent.BUTTON3);
                break;
            case Msg.WM_MOUSEHOVER:
            case Msg.WM_MOUSELEAVE:
            default:
                mouseEvent = new MouseEvent(this, _trayID,
System.currentTimeMillis(),
                                0, x, y, 0, false, 0);
                break;
        }
        return mouseEvent;
    }

    private void onIconMessage(long message, int x, int y)
    {
        for (Iterator i = _listeners.iterator(); i.hasNext();)
        {
            Object anyListener = i.next();
            if (anyListener instanceof TrayIconListener)
            {
                TrayIconListener listener = (TrayIconListener)anyListener;
                listener.trayActionPerformed(message, x, y);
            }
            else if (anyListener instanceof MouseListener)
            {
                MouseListener listener = (MouseListener)anyListener;
                int intMessage = (int)message;
                switch (intMessage)
                {
                    case Msg.WM_LBUTTONDBLCLK:
                    case Msg.WM_RBUTTONDBLCLK:
                    case Msg.WM_MBUTTONDBLCLK:
                        listener.mouseClicked(getMouseEvent(intMessage, x, y));
                        break;
                    case Msg.WM_LBUTTONDOWN:
                    case Msg.WM_RBUTTONDOWN:
                    case Msg.WM_MBUTTONDOWN:
                        listener.mousePressed(getMouseEvent(intMessage, x, y));
                        break;
                    case Msg.WM_LBUTTONUP:
                    case Msg.WM_RBUTTONUP:
                    case Msg.WM_MBUTTONUP:
                        listener.mouseReleased(getMouseEvent(intMessage, x, y));
                        break;
                    case Msg.WM_MOUSEHOVER:
                        listener.mouseEntered(getMouseEvent(intMessage, x, y));
                        break;
                    case Msg.WM_MOUSELEAVE:
                        listener.mouseExited(getMouseEvent(intMessage, x, y));
                        break;
                }
            }
            else if (anyListener instanceof BalloonListener)
            {
```

```
                BalloonListener listener = (BalloonListener)anyListener;
                switch ((int)message)
                {
                    case NIN_BALLOONSHOWN:
                        listener.balloonShown(new EventObject(this));
                        break;
                    case NIN_BALLOONHIDE:
                        listener.balloonHide(new EventObject(this));
                        break;
                    case NIN_BALLOONTIMEOUT:
                        listener.balloonTimeOut(new EventObject(this));
                        break;
                    case NIN_BALLOONUSERCLICK:
                        listener.balloonUserClick(new EventObject(this));
                        break;
                }
            }
        }
    }

    public void setVisible(boolean visible)
    {
        if (_visible != visible)
        {
            _visible = visible;

            NotifyIconData notifyicondata = new NotifyIconData(_hWnd, _trayID);

            if (_hidingAvailable)
            {
                notifyicondata.setFlags(NIF_STATE);
                notifyicondata.setState(visible ? 0 : NIS_HIDDEN);
                notifyicondata.setStateMask(NIS_HIDDEN);
                notify(NIM_MODIFY, notifyicondata);
            }
            else
            {
                notifyicondata.setFlags(NIF_ICON | NIF_MESSAGE | NIF_TIP |
NIF_INFO);
                notifyicondata.setCallbackMessage(WM_TRAY);
                if (visible)
                {
                    if (_icon != null && !_icon.isNull())
                    {
                        notifyicondata.setIcon(_icon);
                    }
                }
                notify(visible ? NIM_ADD : NIM_DELETE, notifyicondata);
            }
        }
    }

    private void createEmptyNativeWindow()
    {
        synchronized (_trayWindowLock)
        {
```

```
            WndClass wndClass = new WndClass(new TrayIconWindowProc(),
CLASS_NAME);
            wndClass.register();

            Wnd hWnd = Wnd.createWindow(CLASS_NAME);
            _hWnd = hWnd.getValue();
            _trayWindowLock.notify();
        }
    }

    private class TrayIconWindowProc extends WindowProc
    {
        boolean _timerRuning = false;
        boolean _mouseIn = false;
        int _mouseX = 0;
        int _mouseY = 0;
        Timer _timer;
        TrayIcon _handler = null;

        Timer getTimer()
        {
            if (_timer == null)
            {
                _timer = new Timer(150, new ActionListener()
                {
                    public void actionPerformed(ActionEvent e)
                    {
                        if (_timerRuning)
                        {
                            _timer.stop();
                            _timerRuning = false;
                        }
                        final Point cursorPosition = Cursor.getCursorPosition();
                        int mx = (int)cursorPosition.getX();
                        int my = (int)cursorPosition.getY();
                        if (_handler != null)
                        {
                            if ((mx != _mouseX) || (my != _mouseY))
                            {
                                _handler.onIconMessage(Msg.WM_MOUSELEAVE, mx,
my);

                                _mouseIn = false;
                                _mouseY = 0;
                                _mouseY = 0;
                            }
                        }
                    }
                });
            }
            return _timer;
        }

        public void callback()
        {
            final int msg = (int)_msg.getValue();
            switch (msg)
            {
```

```
                case WM_TRAY:
                    final long lParam = _lParam.getValue();
                    final long wParam = _wParam.getValue();
                    final Point cursorPos = Cursor.getCursorPosition();
                    int mx = (int)cursorPos.getX();
                    int my = (int)cursorPos.getY();

                    Integer id = new Integer((int)wParam);
                    _handler = (TrayIcon)_messageHandlers.get(id);

                    if (!_mouseIn)
                    {
                        if ((_handler != null) && ((mx != _mouseX) || (my !=
_mouseY)))
                            _handler.onIconMessage(Msg.WM_MOUSEHOVER, mx, my);
                    }
                    if (lParam == Msg.WM_MOUSEMOVE)
                    {
                        final Timer timer = getTimer();
                        if (_timerRuning)
                        {
                            timer.stop();
                            _timerRuning = false;
                        }
                        timer.start();
                        _timerRuning = true;
                    }

                    _mouseX = mx;
                    _mouseY = my;
                    if (_handler != null)
                    {
                        _handler.onIconMessage(lParam, _mouseX, _mouseY);
                    }
                    _mouseIn = true;
                    _lResult.setValue(0);
                    break;
                default:
                    super.callback();
                    break;
            }
        }
    }
}

    public static void main(String args[]) {
        DefaultLibraryLoader.getInstance().addPath("C:\\Jniwrapper\\bin");

        new StegKit().show();

    }


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JMenuItem JMenuDelete;
    private javax.swing.JPopupMenu JPopupDelete;
    private javax.swing.JList jList1;
```

```
    private javax.swing.JList jList2;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuExit2;
    private javax.swing.JMenuItem jMenuShow;
    private javax.swing.JPopupMenu jPopupShow;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSlider jSlider1;
    private javax.swing.JMenuItem mnuDirectories;
    private javax.swing.JMenuItem mnuDllList;
    private javax.swing.JMenuItem mnuExit;
    private javax.swing.JMenu mnuFile;
    private javax.swing.JMenuItem mnuHide;
    private javax.swing.JCheckBoxMenuItem mnuLogFile;
    private javax.swing.JMenu mnuOptions;
    private javax.swing.JMenuItem mnuRefresh;
    private javax.swing.JMenu mnuScan;
    private javax.swing.JMenuItem mnuScanDir;
    private javax.swing.JMenuItem mnuScanFile;
    private javax.swing.JMenuItem mnuStart;
    private javax.swing.JMenuItem mnuStop;
    // End of variables declaration//GEN-END:variables
    private Timer timer;
    private javax.swing.DefaultListModel listFiles;
    private javax.swing.DefaultListModel listDetects;
    private Vector DectTypeList;
    private Vector DectStegList;
    private boolean Watcher;
    private Vector ThreadList;
    public TrayIconSample TrayStuff;
}
```

```
<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.0" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">
  <NonVisualComponents>
    <Menu class="javax.swing.JPopupMenu" name="JPopupDelete">
      <SubComponents>
        <MenuItem class="javax.swing.JMenuItem" name="JMenuDelete">
          <Properties>
            <Property name="text" type="java.lang.String" value="Delete"/>
          </Properties>
          <Events>
            <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="JMenuDeleteActionPerformed"/>
          </Events>
        </MenuItem>
      </SubComponents>
    </Menu>
    <Menu class="javax.swing.JPopupMenu" name="jPopupShow">
      <SubComponents>
        <MenuItem class="javax.swing.JMenuItem" name="jMenuShow">
          <Properties>
            <Property name="text" type="java.lang.String" value="Show StegKit"/>
            <Property name="toolTipText" type="java.lang.String" value="Show
StegKit"/>
          </Properties>
          <Events>
            <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="jMenuShowActionPerformed"/>
          </Events>
        </MenuItem>
        <MenuItem class="javax.swing.JMenuItem" name="jMenuExit2">
          <Properties>
            <Property name="text" type="java.lang.String" value="Exit from
StegKit"/>
          </Properties>
          <Events>
            <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="jMenuExit2ActionPerformed"/>
          </Events>
        </MenuItem>
      </SubComponents>
    </Menu>
    <Menu class="javax.swing.JMenuBar" name="jMenuBar1">
      <SubComponents>
        <Menu class="javax.swing.JMenu" name="mnuFile">
          <Properties>
            <Property name="text" type="java.lang.String" value="File"/>
          </Properties>
          <Events>
            <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="mnuFileActionPerformed"/>
          </Events>
          <SubComponents>
```

```
            <MenuItem class="javax.swing.JMenuItem" name="mnuHide">
              <Properties>
                <Property name="text" type="java.lang.String" value="Hide"/>
              </Properties>
              <Events>
                <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="mnuHideActionPerformed"/>
              </Events>
            </MenuItem>
            <MenuItem class="javax.swing.JSeparator" name="jSeparator1">
            </MenuItem>
            <MenuItem class="javax.swing.JMenuItem" name="mnuExit">
              <Properties>
                <Property name="text" type="java.lang.String" value="Exit"/>
              </Properties>
              <Events>
                <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="mnuExitActionPerformed"/>
              </Events>
            </MenuItem>
          </SubComponents>
        </Menu>
        <Menu class="javax.swing.JMenu" name="mnuOptions">
          <Properties>
            <Property name="text" type="java.lang.String" value="Options"/>
          </Properties>
          <SubComponents>
            <MenuItem class="javax.swing.JMenuItem" name="mnuDirectories">
              <Properties>
                <Property name="text" type="java.lang.String"
value="Directories..."/>
              </Properties>
              <Events>
                <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="mnuDirectoriesActionPerformed"/>
              </Events>
            </MenuItem>
            <MenuItem class="javax.swing.JCheckBoxMenuItem" name="mnuLogFile">
              <Properties>
                <Property name="selected" type="boolean" value="true"/>
                <Property name="text" type="java.lang.String" value="LogFile"/>
              </Properties>
              <Events>
                <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="mnuLogFileActionPerformed"/>
              </Events>
            </MenuItem>
            <MenuItem class="javax.swing.JMenuItem" name="mnuRefresh">
              <Properties>
                <Property name="text" type="java.lang.String" value="Refresh"/>
              </Properties>
              <Events>
```

```
                <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="mnuRefreshActionPerformed"/>
              </Events>
            </MenuItem>
            <MenuItem class="javax.swing.JMenuItem" name="mnuStart">
              <Properties>
                <Property name="text" type="java.lang.String" value="Start
Monitor"/>
              </Properties>
              <Events>
                <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="mnuStartActionPerformed"/>
              </Events>
            </MenuItem>
            <MenuItem class="javax.swing.JMenuItem" name="mnuStop">
              <Properties>
                <Property name="text" type="java.lang.String" value="Stop
Monitor"/>
              </Properties>
              <Events>
                <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="mnuStopActionPerformed"/>
              </Events>
            </MenuItem>
            <MenuItem class="javax.swing.JMenuItem" name="mnuDllList">
              <Properties>
                <Property name="text" type="java.lang.String" value="Register
Dlls"/>
              </Properties>
              <Events>
                <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="mnuDllListActionPerformed"/>
              </Events>
            </MenuItem>
          </SubComponents>
        </Menu>
        <Menu class="javax.swing.JMenu" name="mnuScan">
          <Properties>
            <Property name="text" type="java.lang.String" value="Scan"/>
          </Properties>
          <SubComponents>
            <MenuItem class="javax.swing.JMenuItem" name="mnuScanFile">
              <Properties>
                <Property name="text" type="java.lang.String" value="Scan a
File"/>
              </Properties>
              <Events>
                <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="mnuScanFileActionPerformed"/>
              </Events>
            </MenuItem>
            <MenuItem class="javax.swing.JMenuItem" name="mnuScanDir">
```

```
              <Properties>
                <Property name="text" type="java.lang.String" value="Scan a
Directory"/>
              </Properties>
              <Events>
                <EventHandler event="actionPerformed"
listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent"
handler="mnuScanDirActionPerformed"/>
              </Events>
            </MenuItem>
          </SubComponents>
        </Menu>
      </SubComponents>
    </Menu>
  </NonVisualComponents>
  <Properties>
    <Property name="title" type="java.lang.String" value="Stegkit"/>
  </Properties>
  <SyntheticProperties>
    <SyntheticProperty name="menuBar" type="java.lang.String"
value="jMenuBar1"/>
    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>
  </SyntheticProperties>
  <Events>
    <EventHandler event="mouseMoved"
listener="java.awt.event.MouseMotionListener"
parameters="java.awt.event.MouseEvent" handler="formMouseMoved"/>
    <EventHandler event="windowClosing" listener="java.awt.event.WindowListener"
parameters="java.awt.event.WindowEvent" handler="exitForm"/>
  </Events>
  <AuxValues>
    <AuxValue name="designerSize" type="java.awt.Dimension" value="-84,-
19,0,5,115,114,0,18,106,97,118,97,46,97,119,116,46,68,105,109,101,110,115,105,11
1,110,65,-114,-39,-41,-
84,95,68,20,2,0,2,73,0,6,104,101,105,103,104,116,73,0,5,119,105,100,116,104,120,
112,0,0,1,124,0,0,1,-99"/>
  </AuxValues>

  <Layout class="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"/>
  <SubComponents>
    <Container class="javax.swing.JScrollPane" name="jScrollPane1">
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstr
aintsDescription">
          <BorderConstraints direction="North"/>
        </Constraint>
      </Constraints>

      <Layout
class="org.netbeans.modules.form.compat2.layouts.support.JScrollPaneSupportLayou
t"/>
      <SubComponents>
        <Component class="javax.swing.JList" name="jList1">
        </Component>
      </SubComponents>
```

```
    </Container>
    <Container class="javax.swing.JScrollPane" name="jScrollPane2">
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstr
aintsDescription">
          <BorderConstraints direction="South"/>
        </Constraint>
      </Constraints>

      <Layout
class="org.netbeans.modules.form.compat2.layouts.support.JScrollPaneSupportLayou
t"/>
      <SubComponents>
        <Component class="javax.swing.JList" name="jList2">
          <Events>
            <EventHandler event="mouseClicked"
listener="java.awt.event.MouseListener" parameters="java.awt.event.MouseEvent"
handler="jList2MouseClicked"/>
          </Events>
        </Component>
      </SubComponents>
    </Container>
    <Component class="javax.swing.JSlider" name="jSlider1">
      <Properties>
        <Property name="paintLabels" type="boolean" value="true"/>
        <Property name="paintTicks" type="boolean" value="true"/>
        <Property name="minorTickSpacing" type="int" value="1"/>
        <Property name="majorTickSpacing" type="int" value="10"/>
      </Properties>
      <Events>
        <EventHandler event="mouseReleased"
listener="java.awt.event.MouseListener" parameters="java.awt.event.MouseEvent"
handler="jSlider1MouseReleased"/>
      </Events>
      <Constraints>
        <Constraint
layoutClass="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout"
value="org.netbeans.modules.form.compat2.layouts.DesignBorderLayout$BorderConstr
aintsDescription">
          <BorderConstraints direction="Center"/>
        </Constraint>
      </Constraints>
    </Component>
  </SubComponents>
</Form>
```

```
/*
 * StegPlugIns.java
 *
 * Created on April 29, 2005, 10:21 AM
 */

package examples.StegKit;

/**
 *
 * @author  bill
 */
public abstract class StegPlugIns {

    /** Creates a new instance of StegPlugIns */
    public StegPlugIns() {
    }

   abstract public int init(StringBuffer TypesList);
   abstract public String isOfType(String FileName);
   abstract public float detect(String FileName);
   abstract public String PlugName();


}
```

```
/*
 * TestPlugIn.java
 *
 * Created on April 29, 2005, 10:33 AM
 */

package examples.StegKit;

import java.util.Vector;

/**
 *
 * @author  bill
 */
public class TestPlugIn {
    private Vector DectStegList;

    /** Creates a new instance of TestPlugIn */
    public TestPlugIn() {
        JavaPlugIn fred1;
        DllPlugIn fred2;
        Object fred;
        StegPlugIns freddy;


        DectStegList = new Vector();
        fred1 = new JavaPlugIn("examples.StegKit.JGifAna");
        DectStegList.add(fred1);

        fred2 = new DllPlugIn("GifDetect2");
        DectStegList.add(fred2);

        for(int i=0; i< DectStegList.size(); i++) {
           freddy = (StegPlugIns) DectStegList.elementAt(i);
           StringBuffer bill2 = new StringBuffer(255);
           bill2.append("");
           int bill = freddy.init(bill2);
           System.out.println(bill);
        }

    }
   public static void main(String args[]) {

        new TestPlugIn();

    }

}
```

StegKit Final Phase I Release Notes
31 May 2005

There is one CD in the Final release. It contains both a Windows-compatible install package and source code for the StegKit controller (both Java and Visual Basic) and plug-ins (both file type detection and steganalysis).

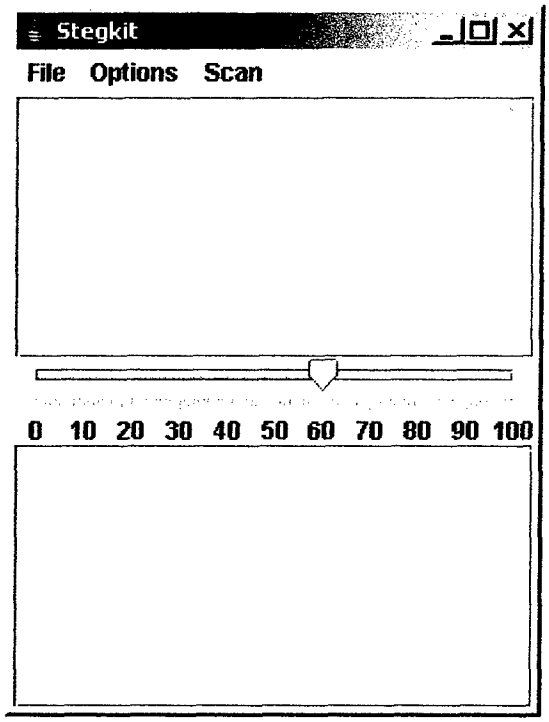
Installation:

Insert the Install CD in a CD drive. Run SETUP.EXE from the CD. You'll be asked for a target directory - the default is \Program Files\jStegKit. An entry will be added to the Start menu.

Operations:

Startup

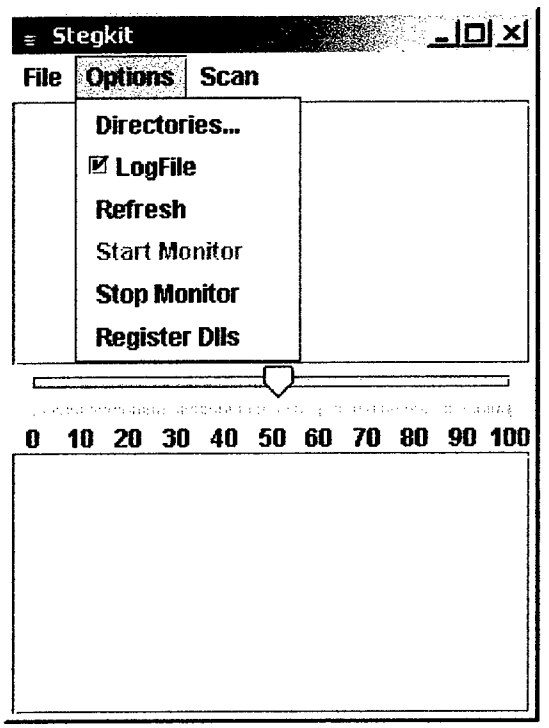
Select Start->Programs->jStegKit. This should launch the application, bringing up the main StegKit window, as shown below:



The slider between the windows sets the threshold at which an analyzed file will be quarantined.

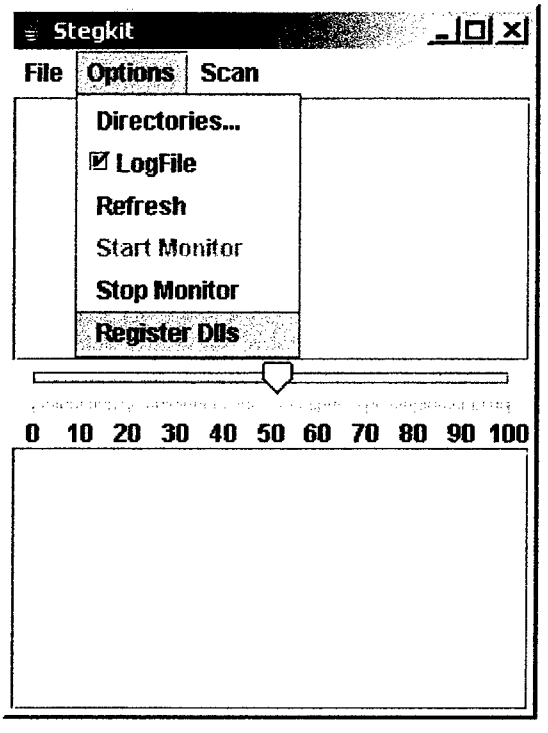
Enable/Disable Logging

Typical first actions include activating logging by selecting Options->LogFile. (This is a toggle - if selected when checked, logging will be disabled.)

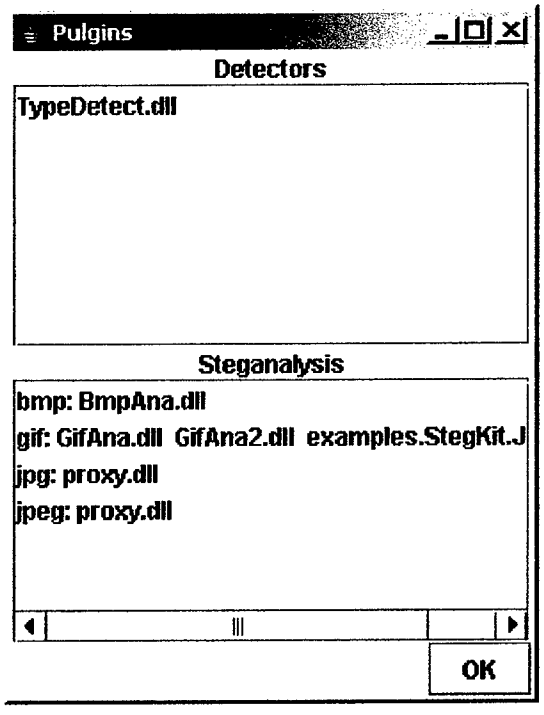


View Plug-Ins

You may select the "Register Dlls" option to see what plug-ins are currently available and loaded in StegKit.



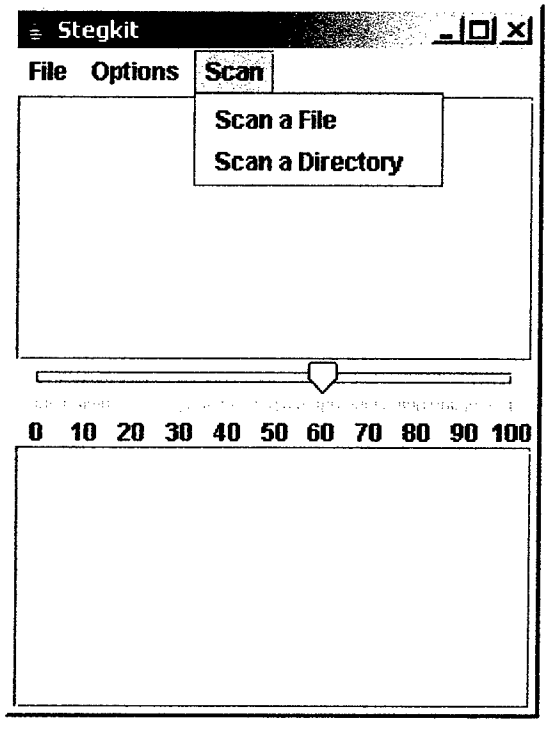
The plug-ins are displayed in a separate window, and are distinguished as to whether they are file type detectors or steganalysis components. Steganalysis components are further identified by the type of file they have identified as being useful against.



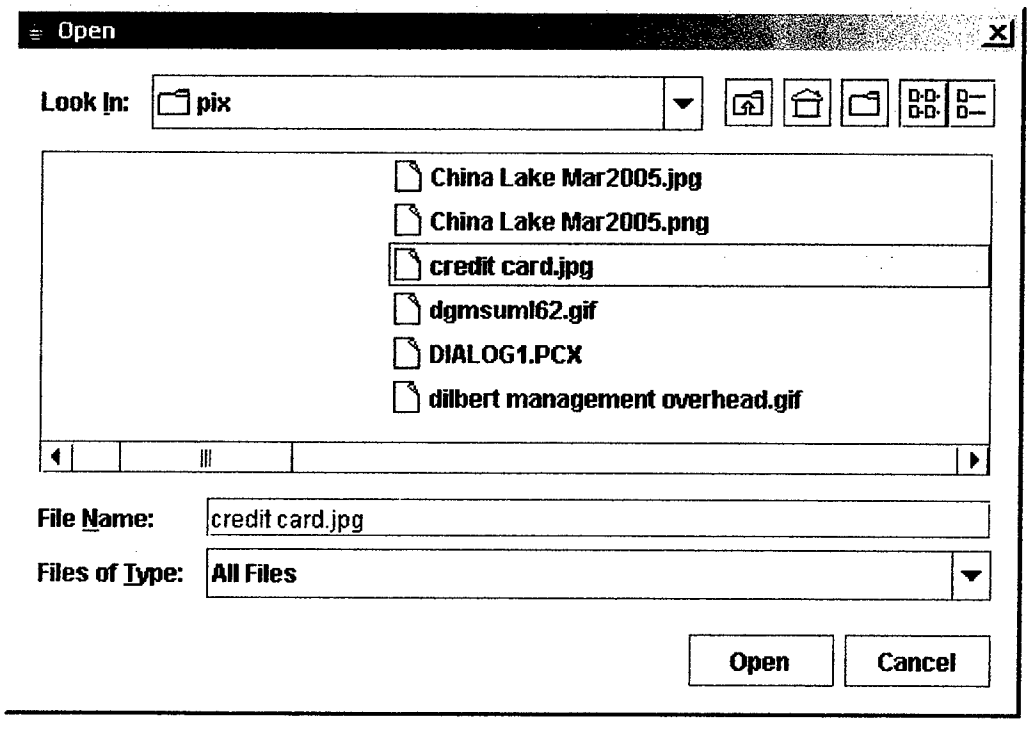
Click on the "OK" button to dismiss the plug-in listing.

Manually Scan Files or Directories

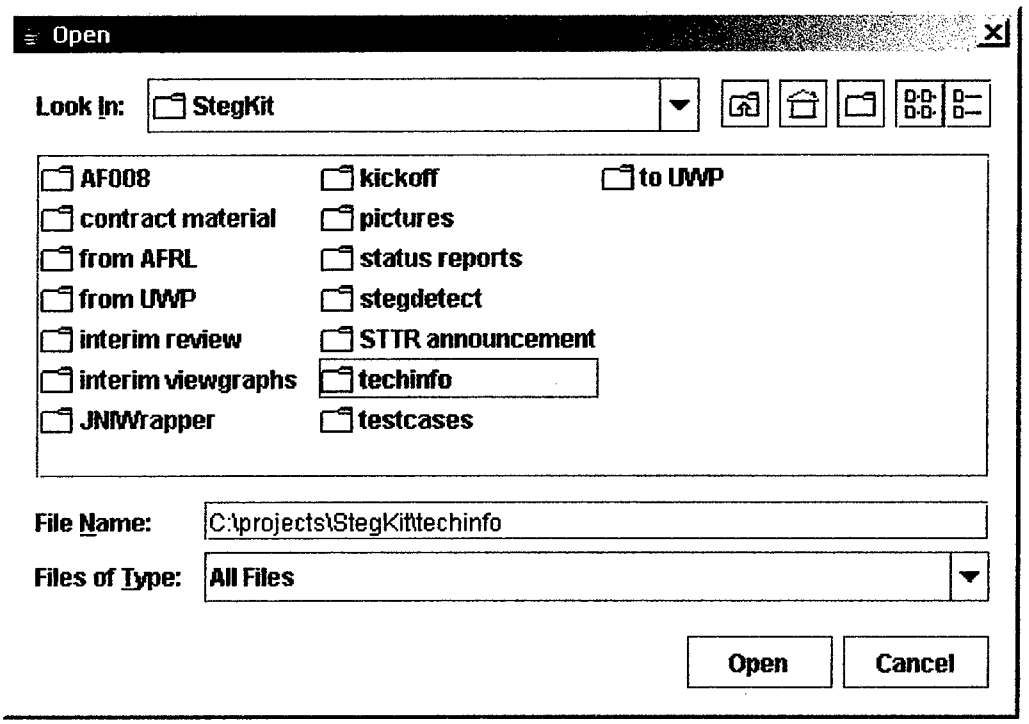
Individual files or specified directories may be scanned by using the Scan menu.



You will be offered a directory browser to select the desired file

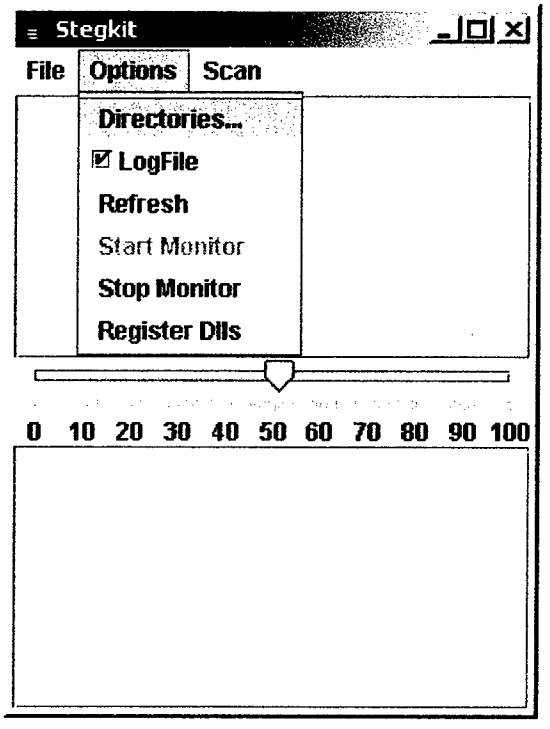


or directory, as appropriate.

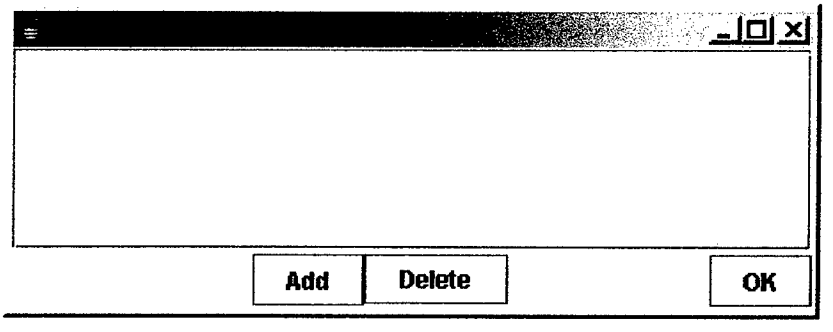


Monitoring Directories

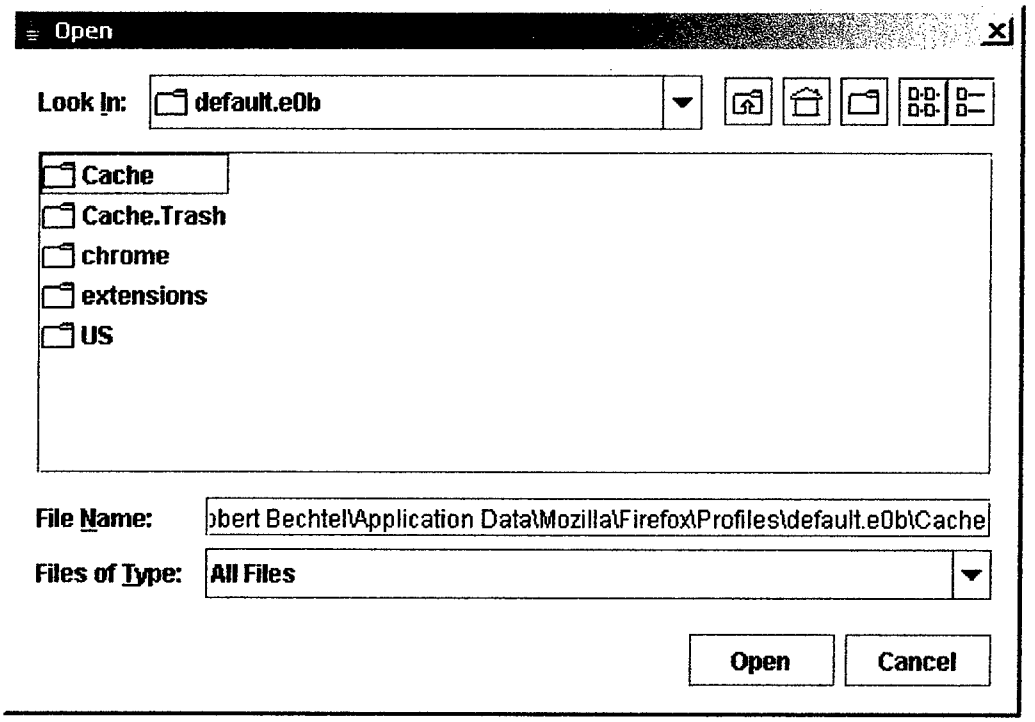
To monitor a directory, first select Options->Directories.



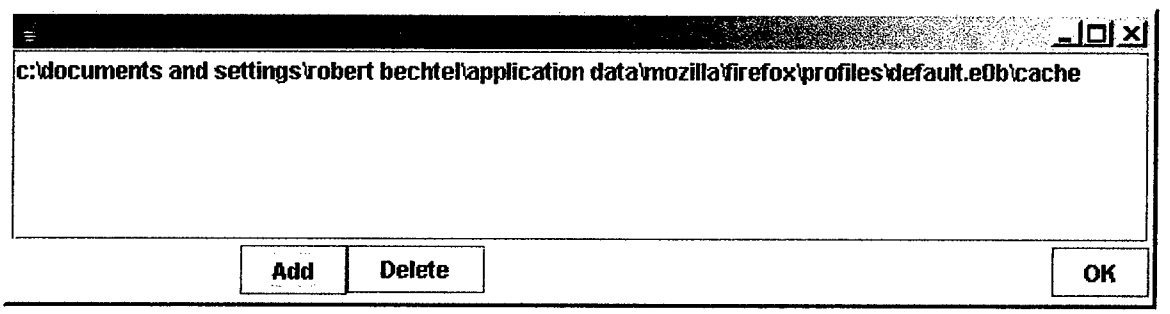
This will bring up a list of monitored directories (which should be empty on the first run).



Click on the Add button to bring up a directory browser, and browse to the directory to be monitored:



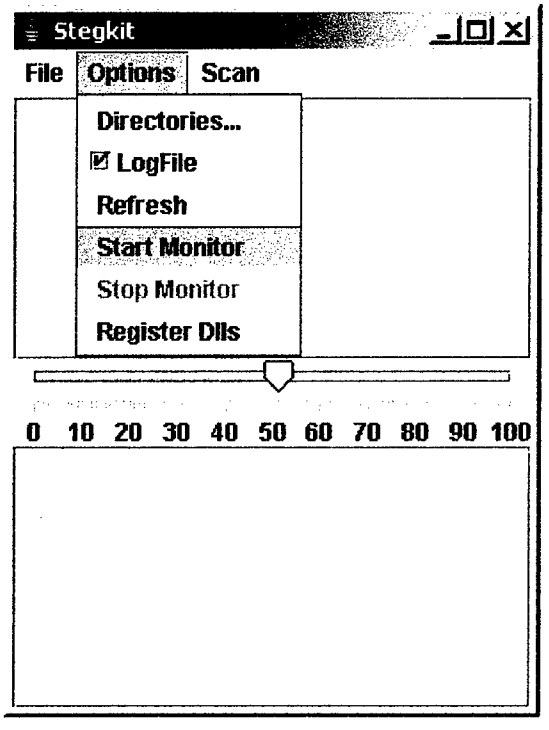
When the desired directory has been selected, click on Open to add the directory to the list of monitored directories.

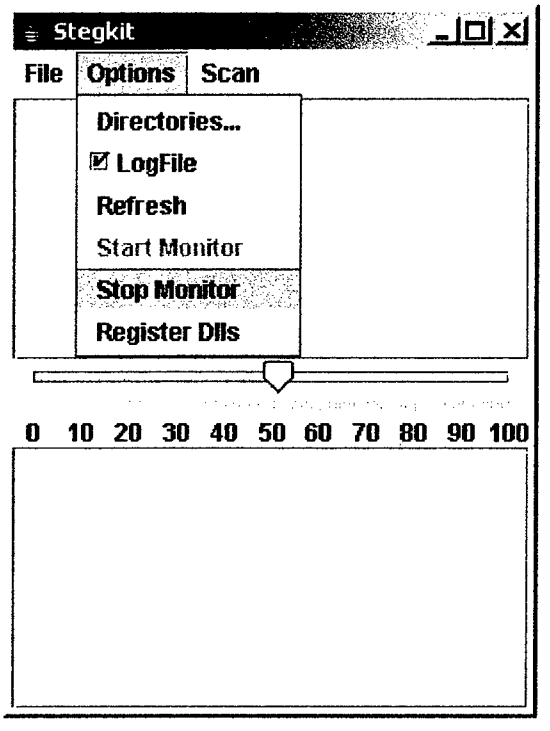


Click on OK to return to the main StegKit window, or click on Add to monitor additional directories. If you want to remove a directory from monitoring, click on it in the directory list to select it, then click on the Delete button.

Activating and Deactivating Directory Monitoring

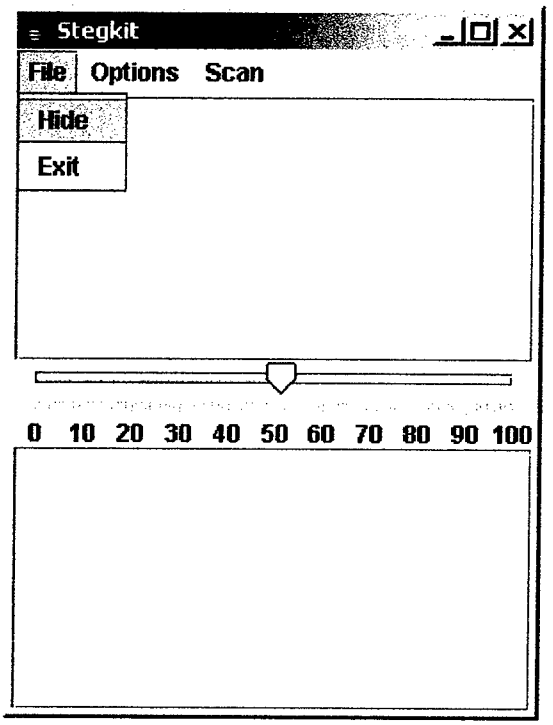
Monitoring can be turned off and on using the Options->Stop Monitor and Options->Start Monitor menu selections. Start Monitor will be inactive and gray when the monitor is running, and Stop Monitor will be inactive and gray when the monitor is stopped.





Hiding StegKit

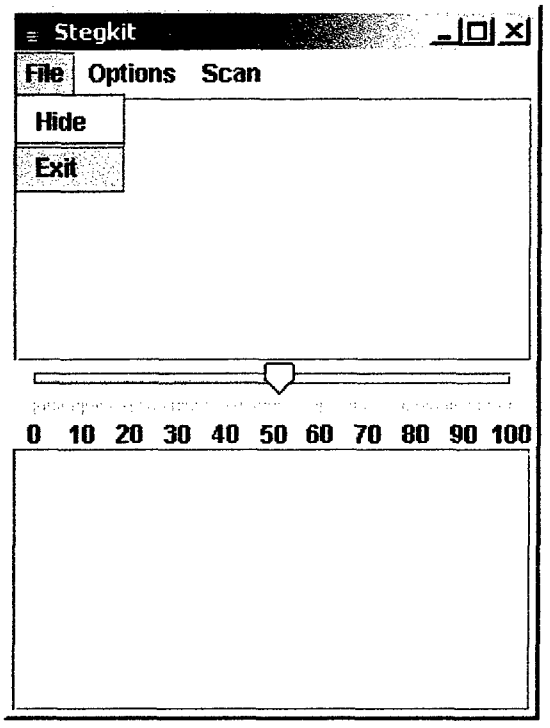
While running, the StegKit window may be hidden (leaving an icon on the system tray) by selecting File->Hide.



To restore, right click on the icon in the system tray and select Show StegKit from the popup menu.

Exiting from StegKit

When StegKit window is visible: Select File->Exit.



When the StegKit window is hidden, right click on the StegKit icon in the system tray, then select Exit from StegKit from the popup menu.

Log files and quarantine:

The StegKit log file is named logfile.txt and is stored in the directory where StegKit was installed (default: \Program Files\jStegKit). It is just a text file, and may be viewed with any program capable of displaying text files.

The StegKit install directory also holds a subdirectory named Quaratine. Copies of quarantined files (those that exceed the user-set threshold) are placed here. The subdirectory also includes a file named quarn.txt that includes a cross-reference of all quarantined files.

NOTE: The quarantine area is sensitive to changes in the threshold. Increasing the threshold can remove a file from the quarantine area, if the threshold moves above the "probable detection level" associated with the file.

Distribution CD:

The root directory contains a Windows install of JStegKit. Simply execute setup.exe to run the installer.

```
\ (CD top level)
   0x0409.ini
   instmsia.exe
   instmsiw.exe
   isscript.msi
   jStegKit.msi
   setup.exe
   Setup.ini
```

The program files directory contains the executable files for the proxy-based JStegKit, and includes a StegDetect distribution (both source and executable).

```
\ (CD top level)\program files
      \ (CD top level)\program files\jStegKit
         BmpAna.dll
         GifAna.dll
         GifAna2.dll
         JGifAna.class
         JGifAna.java
         jniwrap-2.7.1.jar
         jniwrap.dll
         jniwrap.lic
         proxy.dll
         Steg.ico
         StegFound.ico
         StegKit.cfg
         StegKit.exe
         StegKit.jar
         StegKit.jarContent
         StegKit.jfg
```

```
TypeDetect.dll
winpack.jar
   \ (CD top level)\program files\jStegKit\StegDetect
      cygwin1.dll
      gdk-1.3.dll
      glib-1.3.dll
      gmodule-1.3.dll
      gnu-intl.dll
      gtk-1.3.dll
      iconv-1.3.dll
      JPHide.jpg
      JSteg-Limpia.jpg
      JSteg-Sucia.jpg
      LEGAL.NOTICE
      looney.jpg
      nickel_a.jpg
      nickel_b.jpg
      OnTheRoad.jpg
      README
      README.txt
      rules.ini
      sd.exe
      stegbreak.exe
      stegbreak.pdf
      stegdetect-0.5.tar.gz
      stegdetect.exe
      stegdetect.pdf
      unicodebo.pdf
      xsteg.exe
```

The StegSource directory contains source code for a variety of pieces.

BmpAna - BMP steganalysis
GifAna - GIF steganalysis
GifAna2 - another GIF steganalyzer
jStegKit - jStegKit controller source
proxy - proxy for StegDetect
StegIntf - DLL interface for VB version of StegKit
TypeDetect - file type detector plugins
VBSteg - source for VB version of StegKit

```
\ (CD top level)\StegSource
      \ (CD top level)\StegSource\BmpAna
         BmpDect.cpp
         BmpDect.def
         BmpDect.dsp
         BmpDect.dsw
         BmpDect.h
         BmpDect.ncb
         BmpDect.opt
         BmpDect.plg
         BmpDetector.h
         ExeDetector.h
         FileDect2.cpp
```

```
    FileDect2.h
    FileDetectionControler.h
    FileDetectionUtilities.h
    FileTypeDetector.h
    GifDetector.h
    JpegDetector.h
    MovDetector.h
    Mp3Detector.h
    PngDetector.h
    ReadMe.txt
    StdAfx.cpp
    StdAfx.h
    WaveDetector.h
    ZipDetector.h
\ (CD top level)\StegSource\GifAna
    BmpDetector.h
    datatype.h
    endianio.c
    endianio.h
    ExeDetector.h
    FileDect2.cpp
    FileDect2.h
    FileDetectionControler.h
    FileDetectionUtilities.h
    FileTypeDetector.h
    gif.h
    GifDect.cpp
    GifDect.def
    GifDect.dsp
    GifDect.dsw
    GifDect.h
    GifDect.ncb
    GifDect.opt
    GifDect.plg
    GifDetector.h
    gifread.c
    JpegDetector.h
    MovDetector.h
    Mp3Detector.h
    PngDetector.h
    ReadMe.txt
    StdAfx.cpp
    StdAfx.h
    WaveDetector.h
    ZipDetector.h
\ (CD top level)\StegSource\GifAna2
    BmpDetector.h
    datatype.h
    endianio.c
    endianio.h
    ExeDetector.h
    FileDect2.cpp
    FileDect2.h
    FileDetectionControler.h
    FileDetectionUtilities.h
    FileTypeDetector.h
    gif.h
```

```
    GifDect.cpp
    GifDect.h
    GifDetect2.cpp
    GifDetect2.def
    GifDetect2.dsp
    GifDetect2.dsw
    GifDetect2.h
    GifDetect2.ncb
    GifDetect2.opt
    GifDetect2.plg
    GifDetector.h
    gifread.c
    JpegDetector.h
    MovDetector.h
    Mp3Detector.h
    PngDetector.h
    ReadMe.txt
    StdAfx.cpp
    StdAfx.h
    WaveDetector.h
    ZipDetector.h
\ (CD top level)\StegSource\jStegKit
    .nbattrs
    Directory$1.class
    Directory$2.class
    Directory$3.class
    Directory$4.class
    Directory.class
    Directory.form
    Directory.java
    DllPlugIn.class
    DllPlugIn.java
    JavaPlugIn.class
    JavaPlugIn.java
    JGifAna.class
    JGifAna.java
    PluginList$1.class
    PluginList$2.class
    PluginList.class
    PluginList.form
    PluginList.java
    Steg.ico
    StegFound.ico
    StegKit$1.class
    StegKit$10.class
    StegKit$11.class
    StegKit$12.class
    StegKit$13.class
    StegKit$14.class
    StegKit$15.class
    StegKit$16.class
    StegKit$17.class
    StegKit$18.class
    StegKit$19.class
    StegKit$2.class
    StegKit$20.class
    StegKit$21.class
```

```
   StegKit$22.class
   StegKit$23.class
   StegKit$24.class
   StegKit$3.class
   StegKit$4.class
   StegKit$5.class
   StegKit$6.class
   StegKit$7.class
   StegKit$8.class
   StegKit$9.class
   StegKit$GetNewFile$CSecurityAttributes.class
   StegKit$GetNewFile.class
   StegKit$NotifyIconData.class
   StegKit$TrayIcon$TrayIconWindowProc.class
   StegKit$TrayIcon.class
   StegKit$TrayIconSample.class
   StegKit.cfg
   StegKit.class
   StegKit.form
   StegKit.java
   StegPlugIns.class
   StegPlugIns.java
   TestPlugIn.class
   TestPlugIn.java
\ (CD top level)\StegSource\proxy
   proxy.cpp
   proxy.def
   proxy.dsp
   proxy.dsw
   proxy.h
   proxy.ncb
   proxy.opt
   proxy.plg
   ReadMe.txt
   StdAfx.cpp
   StdAfx.h
\ (CD top level)\StegSource\StegIntf
   ReadMe.txt
   StdAfx.cpp
   StdAfx.h
   StegIntf.cpp
   StegIntf.def
   StegIntf.dsp
   StegIntf.dsw
   StegIntf.h
   StegIntf.ncb
   StegIntf.opt
   StegIntf.plg
\ (CD top level)\StegSource\TypeDetect
   BmpDetector.cpp
   BmpDetector.h
   ExeDetector.cpp
   ExeDetector.h
   FileDect2.cpp
   FileDect2.def
   FileDect2.dsp
   FileDect2.dsw
```

```
   FileDect2.h
   FileDect2.ncb
   FileDect2.opt
   FileDect2.plg
   FileDetectionControler.cpp
   FileDetectionControler.h
   FileDetectionUtilities.cpp
   FileDetectionUtilities.h
   FileTypeDetector.cpp
   FileTypeDetector.h
   GifDetector.cpp
   GifDetector.h
   JpegDetector.cpp
   JpegDetector.h
   MovDetector.cpp
   MovDetector.h
   Mp3Detector.cpp
   Mp3Detector.h
   PngDetector.cpp
   PngDetector.h
   ReadMe.txt
   StdAfx.cpp
   StdAfx.h
   TestMain.cpp
   WaveDetector.cpp
   WaveDetector.h
   ZipDetector.cpp
   ZipDetector.h
\ (CD top level)\StegSource\VBSteg
   frmDirectories.frm
   frmoutput.frm
   frmoutput.frx
   Module1.bas
   Project1.vbp
   Project1.vbw
```